APPLICATION OF EIGENVALUE SENSITIVITY AND EIGENVECTOR SENSITIVITY IN EIGENCOMPUTATIONS

BY

PURANDAR SARMAH

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1993

ACKNOWLEDGEMENTS

I am indebted to a number of people, directly or indirectly, for helping me to prepare my dissertation. First, I would like to thank my advisor Dr. William Hager for his advice and help throughout the different stages of this dissertation. I am grateful to Dr. Kermit Sigmon and Dr. James Keesling for working with me on several algorithms which I used to get some numerical results in my dissertation. I would like to thank Dr. Bruce Edwards for his invaluable advice as my supervisory committee member. Without his advice I would not have been able to push my work this far. I am grateful to Dr. Nilotpal Ghosh, who was a visiting professor in our department. His work on eigenvalue problem had direct influence on some of my results.

Finally, I want to express my deep sense of gratitude to my wife for helping me in different ways. Without her encouragement and help during the more frustrating moments of my research, this work would not have come to this final stage.

TABLE OF CONTENTS

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

APPLICATION OF EIGENVALUE SENSITIVITY AND EIGENVECTOR SENSITIVITY IN EIGENCOMPUTATIONS

By

PURANDAR SARMAH

December 1993

Chairman: William Hager
Major Department: Mathematics

In 1987, in his paper "Bidiagonalization and Diagonalization," William W. Hager presented an algorithm to diagonalize a matrix with distinct eigenvalues. To implement the algorithm, we need a good approximation to the matrix of eigenvectors. First, we present the derivation of his algorithm, and then we show how it can be generalized to diagonalize a matrix with multiple eigenvalues. A local quadratic convergence result is established for the generalized algorithm. When a good starting guess for the eigenvectors is not known, Hager's algorithm may not converge. To restore convergence, we modified the algorithm by replacing diagonalizations by Schur decompositions. We found that if uniformly distributed numbers on a circle in the complex plane, which includes all Gerschgorin disks for the matrix, are used to approximate the eigenvalues, and if the eigenvectors are approximated by the columns of the identity matrix, then rapid convergence is obtained. We presented some node programs which, along with some modifications to the last algorithm, can be used in a distributed memory machine, where processors are interconnected in a ring. Next,

we present additional modifications of Hager's algorithm and the generalized algorithm that take advantage of matrix symmetry and reduce the given matrix into a diagonal matrix with eigenvalues along the diagonal.

CHAPTER 1
INTRODUCTION

Let A be an $n \times n$ matrix. We want to find an iterative method to compute a diagonalization $A = X\Lambda X^{-1}$, provided it exists, where the columns of X are eigenvectors of A, and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ whose diagonal elements are the eigenvalues of A. Here we will use a sensitivity result for eigenpairs to diagonalize a matrix. We assume eigenvalues of A are all distinct. In the derivation of necessary equations to diagonalize a matrix, we need the continuity of eigenvalues, and the perturbation theory based on Gerschgorin's theorem. We give the statement of the theorem below:

Theorem 1.0.1 *If* $A \in \mathbb{C}^{n\times n}$, $\rho_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|$, *and*

$$D_i = \{z \in \mathbb{C} : |z - a_{ii}| < \rho_i\},$$

then

$$\lambda(A) \subseteq \cup_{i=1}^{n} D_i.$$

Furthermore, if a set of k *Gerschgorin disks* D_i *are isolated from the other* $n - k$ *disks, then their union contains precisely* k *eigenvalues of* A.

Proof of Theorem 1.0.1 For proof, the interested readers are referred to [Atk89, Theorem 9.1, page 588].

Let A be an $n \times n$ matrix, whose eigenvalues are all distinct, and let $X^{-1}AX = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_j, \ldots, \lambda_n)$ be a diagonalization of A. Then $Ax_j = \lambda_j x_j$, where

x_j is the jth column of X. Given an arbitrary matrix E, and for a sufficiently small number ϵ, we will show in the next theorem that there exist continuously differentiable functions $\lambda_j(\epsilon)$, and $x_j(\epsilon)$ with $\lambda_j(0) = \lambda_j$, and $x_j(0) = x_j$ such that $(A + \epsilon E)x_j(\epsilon) = \lambda_j(\epsilon)x_j(\epsilon)$. That is $(\lambda_j(\epsilon), x_j(\epsilon))$ is an eigenpair of $A + \epsilon E$.

Theorem 1.0.2 *Let A be an $n \times n$ matrix. Assume the eigenvalues of A are all distinct. Suppose $X^{-1}AX = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a diagonalization of A. Then given an arbitrary $n \times n$ matrix E, and for a sufficiently small ϵ, there exist continuously differentiable functions $\lambda_j(\epsilon)$, and $x_j(\epsilon)$ with $\lambda_j(0) = \lambda_j$, and $x_j(0) = x_j$ such that the following matrix equation holds:*

$$(A + \epsilon E)x_j(\epsilon) = \lambda_j(\epsilon)x_j(\epsilon). \tag{1.1}$$

Proof of Theorem 1.0.2 Let $X^{-1}AX = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_j, \ldots, \lambda_n)$ be a diagonalization of A. Let E be an $n \times n$ matrix, and ϵ be a small number. Consider the matrix $D = X^{-1}(A + \epsilon E)X$. Then

$$d_{ij} = \begin{cases} \lambda_j + \epsilon f_{jj} & \text{for } i = j, \\ \epsilon f_{ij} & \text{for } i \neq j, \end{cases}$$

where $f_{ij} = y_i^T E x_j$, y_i^T is the ith row of X^{-1}. Let $B = \text{diag}(1, 1, \ldots, \epsilon/k, \ldots, 1)$, and let $C = BDB^{-1}$, then

$$C = \begin{bmatrix} \lambda_1 & & & & & \\ & \lambda_2 & & & & \\ & & \ddots & & & \\ & & & \lambda_j & & \\ & & & & \ddots & \\ & & & & & \lambda_n \end{bmatrix} + \begin{bmatrix} \epsilon f_{11} & \epsilon f_{12} & \cdots & kf_{1j} & \cdots & \epsilon f_{1n} \\ \epsilon f_{21} & \epsilon f_{22} & \cdots & kf_{2j} & \cdots & \epsilon f_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ \frac{\epsilon^2}{k} f_{j1} & \frac{\epsilon^2}{k} f_{j2} & \cdots & \epsilon f_{jj} & \cdots & \frac{\epsilon^2}{k} f_{jn} \\ \vdots & \vdots & & \vdots & & \vdots \\ \epsilon f_{n1} & \epsilon f_{n2} & \cdots & kf_{nj} & \cdots & \epsilon f_{nn} \end{bmatrix}.$$

Applying Theorem 1.0.1 to C, we find that the jth eigenvalue lies in the circular disk with center $(\lambda_j + \epsilon f_{jj})$, and radius $r_j = \dfrac{\epsilon^2}{k} \displaystyle\sum_{\substack{i=1 \\ i \neq j}}^{n} |f_{ji}|$, provided ϵ is sufficiently small,

and k is chosen so as to make the jth Gerschgorin disk disjoint from the other $n-1$ disks. We may choose a value of k, which is independent of ϵ in the following way. Choose k to have the largest value consistent with the inequalities:

$$|kf_{ij}| \leq \frac{1}{2}|\lambda_j - \lambda_i|,\ i \neq j,\ i = 1, \ldots, n.$$

We achieve the above inequality if

$$k = \min_{i \neq j} \left| \frac{\lambda_j - \lambda_i}{2f_{ij}} \right|,\ i \neq j,\ i = 1, \ldots, n.$$

With the above value of k, the jth disk will be isolated from the other $n-1$ disks, the radius of the jth disk will be of order ϵ^2, and the perturbation in the jth eigenvalue will be of order ϵ as $\epsilon \to 0$.

Next we discuss the effect of perturbations in the elements of A on the eigenvectors in its diagonalization. As earlier, let $X^{-1}AX = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_j, \ldots, \lambda_n)$ be a diagonalization of A. The columns $x_1, x_2, \ldots, x_n$ of X form a complete set of eigenvectors. We denote eigenvectors of $A + \epsilon E$ by $x_1(\epsilon), x_2(\epsilon), \ldots, x_n(\epsilon)$, and eigenvectors of $D = X^{-1}(A + \epsilon E)X$ by $v_1(\epsilon), v_2(\epsilon), \ldots, v_n(\epsilon)$ such that $x_i(\epsilon) = Xv_i(\epsilon)$. Since $(\lambda_j(\epsilon), v_j(\epsilon))$ is an eigenpair of D, and $\lambda_j(0) = \lambda_j$, so we must have $v_j(0) = e_j$, which is the jth unit vector. Let $v_j(\epsilon)$ be normalized so that its largest element is 1. For sufficiently small ϵ, claim that the jth element of $v_j(\epsilon)$ is 1. If not, let $v_{ij}(\epsilon) = 1$ for some i, $i \neq j$. Since $Dv_j(\epsilon) = \lambda_j(\epsilon)v_j(\epsilon)$, so equating the ith component on both sides, we get

$$\lambda_j(\epsilon)v_{ij}(\epsilon) = \lambda_i v_{ij}(\epsilon) + \epsilon \sum_{l=1}^{n} f_{il}v_{lj}(\epsilon), \tag{1.2}$$

that is

$$\lambda_j(\epsilon) - \lambda_i = \epsilon \sum_{l=1}^{n} f_{il}v_{lj}(\epsilon).$$

Now letting $\epsilon \to 0$, we find $\lambda_j - \lambda_i = 0$, $i \neq j$, a contradiction. So for sufficiently small ϵ, $v_{jj}(\epsilon) = 1$. Next, we will show that the other components of $v_j(\epsilon)$ are all

less than $N\epsilon$ as $\epsilon \to 0$ for some $N > 0$. Since $|v_{ij}(\epsilon)| \leq 1$, so from (1.2), we get

$$|\lambda_j(\epsilon) - \lambda_i||v_{ij}(\epsilon)| \leq \epsilon \sum_{l=1}^{n} |f_{il}|.$$

Let $|\lambda_j(\epsilon) - \lambda_i| \geq \frac{1}{2}|\lambda_j - \lambda_i|$ for $i \neq j$, $i = 1, \ldots, n$, then $|v_{ij}(\epsilon)| \leq N\epsilon$, where

$$N = \max_{i \neq j} \left\{ \frac{2\sum_{l=1}^{n} |f_{il}|}{|\lambda_j - \lambda_i|} \right\}.$$

Thus $v_{ij}(\epsilon)$ are of order ϵ for $i \neq j$, $i = 1, \ldots, n$. Also from (1.2), we get

$$(\lambda_j(\epsilon) - \lambda_i)v_{ij}(\epsilon) = \epsilon f_{ij} + \epsilon \sum_{\substack{l=1 \\ l \neq i}}^{n} f_{il} v_{lj}(\epsilon).$$

The second term on the right is of order ϵ^2, so for $i \neq j$, $i = 1, \ldots, n$, we have

$$\left| v_{ij}(\epsilon) - \frac{\epsilon f_{ij}}{\lambda_j - \lambda_i} \right| = O(\epsilon^2).$$

Hence $\lambda_j(\epsilon)$, and $x_j(\epsilon)$ are differentiable functions of ϵ, and their first derivatives are continuous. This completes the proof of the theorem. □

Now we are ready to derive the necessary equations to find a diagonalization of a matrix A, whose eigenvalues are all distinct.

Let (λ_j, x_j) be a simple eigenvalue, and corresponding eigenvector of A. Given an $n \times n$ matrix E, and for a sufficiently small ϵ, let $\lambda_j(\epsilon)$, and $x_j(\epsilon)$ be differentiable functions of ϵ such that $\lambda_j(0) = \lambda_j$, and $x_j(0) = x_j$. Differentiating (1.1) with respect to ϵ and putting $\epsilon = 0$, we have

$$Ax_j'(0) + Ex_j = \lambda_j'(0)x_j + \lambda_j x_j'(0). \tag{1.3}$$

Since the columns $x_1, x_2, \ldots, x_n$ of X form a complete set of eigenvectors of A, we can expand $x_j(\epsilon)$ in the following way:

$$x_j(\epsilon) = \sum_{i=1}^{n} b_{ij}(\epsilon)x_i. \tag{1.4}$$

Since a scalar multiple of an eigenvector is again an eigenvector, we can take $b_{jj}(\epsilon) = 1$ without loss of generality.

Now differentiating (1.4) with respect to ϵ and setting $\epsilon = 0$, we have

$$x_j'(0) = \sum_{\substack{i=1 \\ i \neq j}}^{n} b_{ij}'(0) x_i. \tag{1.5}$$

Substituting (1.5) into (1.3), we obtain

$$A \sum_{\substack{i=1 \\ i \neq j}}^{n} b_{ij}'(0) x_i + E x_j = \lambda_j'(0) x_j + \lambda_j \sum_{\substack{i=1 \\ i \neq j}}^{n} b_{ij}'(0) x_i$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} b_{ij}'(0) \lambda_i x_i + E x_j = \lambda_j'(0) x_j + \sum_{\substack{i=1 \\ i \neq j}}^{n} b_{ij}'(0) \lambda_j x_i$$

$$E x_j = \lambda_j'(0) x_j + \sum_{\substack{i=1 \\ i \neq j}}^{n} b_{ij}'(0) \left(\lambda_j - \lambda_i\right) x_i. \tag{1.6}$$

To obtain expressions for $\lambda_j'(0)$, and $b_{ij}'(0)$ we use relations

$$y_i^T x_k = \begin{cases} 0 & \text{for } i \neq k, \\ 1 & \text{for } i = k, \end{cases} \tag{1.7}$$

where y_i^T is the ith row of X^{-1}. So premultiplying (1.6) by y_j^T, and y_k^T, yields

$$\lambda_j'(0) = y_j^T E x_j. \tag{1.8}$$

$$y_k^T E x_j = b_{kj}'(0)(\lambda_j - \lambda_k) \quad \Longrightarrow \quad b_{kj}'(0) = \frac{y_k^T E x_j}{\lambda_j - \lambda_k}.$$

With these values of $b_{kj}'(0)$, (1.5) becomes

$$x_j'(0) = \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{y_i^T E x_j}{\lambda_j - \lambda_i} x_i. \tag{1.9}$$

A first order Taylor expansion gives

$$\begin{aligned} \lambda_j(\epsilon) &\approx \lambda_j(0) + \epsilon \lambda_j'(0), \\ x_j(\epsilon) &\approx x_j(0) + \epsilon x_j'(0). \end{aligned}$$

Substituting the values of $\lambda_j'(0)$, and $x_j'(0)$ from (1.8), and (1.9) in the above approximations, we get

$$\begin{aligned} \lambda_j(\epsilon) &\approx \lambda_j(0) + \epsilon y_j^T E x_j, \\ x_j(\epsilon) &\approx x_j(0) + \epsilon \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{y_i^T E x_j}{\lambda_j - \lambda_i} x_i. \end{aligned} \tag{1.10}$$

To develop an algorithm we use (1.10) in the following way. Suppose $X\Lambda X^{-1}$ is an approximate diagonalization of B. If we identify E in (1.10) with $B - X\Lambda X^{-1}$, then to the first order:

$$\begin{aligned} \lambda_j(\epsilon) &\approx \lambda_j(0) + \epsilon y_j^T \left(B - X\Lambda X^{-1} \right) x_j \\ &\approx \lambda_j(0) + \epsilon y_j^T B x_j - \epsilon \lambda_j(0), \\ x_j(\epsilon) &\approx x_j(0) + \epsilon \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{y_i^T \left(B - X\Lambda X^{-1} \right) x_j}{\lambda_j - \lambda_i} x_i \\ &\approx x_j(0) + \epsilon \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{y_i^T B x_j}{\lambda_j - \lambda_i} x_i. \end{aligned}$$

With $\epsilon = 1$, these approximations for an eigenvalue, and corresponding eigenvector give:

$$\begin{aligned} \lambda_j^{new} &= \left(y_j^{old} \right)^T B x_j^{old} && \text{for } j = 1:n, \\ x_j^{new} &= x_j^{old} + \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{\left(y_i^{old} \right)^T B x_j^{old}}{\lambda_j^{new} - \lambda_i^{new}} x_i^{old} && \text{for } j = 1:n, \\ Y^{new} &= \left(X^{new} \right)^{-1}. \end{aligned} \tag{1.11}$$

We can use the above expressions to formulate an algorithm to compute a diagonalization of a nondefective matrix A as follows:

Algorithm 1.0.1 (Diagonalization) Given a nondefective matrix $A \in \mathbb{C}^{n \times n}$, a matrix of approximate eigenvectors X_0, and a tolerance *tol* greater than the unit roundoff, the following algorithm computes a diagonalization $A = X\Lambda X^{-1}$.

Dif = 1; $X = X_0 = [\, x_1, x_2, \ldots, x_n \,]$; $Y = X^{-1} = [\, y_1, y_2, \ldots, y_n \,]^T$

until Dif < *tol*

$C = 0$; { $C = [\, c_1, c_2, \ldots, c_n \,]$ is a $n \times n$ zero matrix. }

for $j = 1 : n$

$\lambda_j \leftarrow (y_j)^T A x_j$

end

for $j = 1 : n$

for $i = 1 : n$

if $i \neq j$

$$c_j \leftarrow c_j + \frac{(y_i)^T A x_j}{\lambda_j - \lambda_i} x_i$$

end

end

$x_j \leftarrow x_j + c_j$; $x_j \leftarrow x_j / ||x_j||$

end

$Y \leftarrow X^{-1}$

Dif $\leftarrow ||C||_F$; { where $|| \cdot ||_F$ is the Frobenious matrix norm. }

end

Above, we use a normalization for x_j to reduce the growth of X.

Example 1.0.1 If Algorithm 1.0.1 is applied to

$$A = \begin{bmatrix} 8.0 & -1.4 & 0.8 & 6.5 & 1.8 \\ 2.3 & 7.1 & 1.3 & 1.7 & 5.0 \\ -3.1 & -4.7 & -0.1 & -2.4 & -1.3 \\ -4.5 & 3.8 & -2.3 & -7.7 & 0.2 \\ 1.2 & 0.7 & -0.5 & -0.2 & -1.4 \end{bmatrix}$$

with $X_0 = I$, and $tol = 10^{-9}$, then the eigenvalues converge as shown in the following table:

Iteration	λ_1	λ_2	λ_3	λ_4	λ_5
1	8.0000	7.1000	−0.1000	−7.7000	−1.4000
2	7.6170	5.0012	0.9699	−6.7182	−0.9700
3	7.1096	5.4788	1.5900	−6.1843	−2.0941
4	8.7039	4.5657	0.9711	−6.0978	−2.2429
5	7.9836	5.0217	0.8616	−6.0530	−1.9139
6	7.9953	4.9963	0.8889	−6.0571	−1.9235
7	7.9943	4.9964	0.8892	−6.0571	−1.9229
8	7.9943	4.9964	0.8892	−6.0571	−1.9229

The exponents of convergence p in $\frac{||X_{k+1} - X||_F^p}{||X_k - X||_F}$ for different values of k are given in the following table, where X_k is the matrix of eigenvectors at the kth iteration, and X is the final matrix of eigenvectors:

k	p
1	3.207
2	0.430
3	4.123
4	2.086
5	2.104
6	1.962

Hence the method appears to converge quadratically. (N. Ghosh gave a theoretical proof of the quadratic convergence of the diagonalization method).

Example 1.0.2 If Algorithm 1.0.1 is applied to

$$A = \begin{bmatrix} 55 & 7 & 61 & 51 & 79 \\ 47 & 52 & 21 & 61 & 58 \\ 74 & 33 & 4 & 79 & 60 \\ 68 & 56 & 59 & 29 & 52 \\ 42 & 73 & 26 & 20 & 6 \end{bmatrix}$$

with $X_0 = I$, and $tol = 10^{-8}$, then the approximate eigenvalues do not converge to the eigenvalues of A, which is clear from the following table:

Iteration	λ_1	λ_2	λ_3	λ_4	λ_5
1	55	52	4	29	6
100	−1241.77	−4853.27	1586.28	5100.90	−446.15
200	124807.75	−318959.32	−368210.46	369633.58	192874.49
300	416577.75	230821	−768266.74	−1252454	1373474
400	539592	−163324.75	474518.5	−868096	17506.02
500	685138	422251	17628	−235428.5	−889446
600	−13438605.19	−1459224.5	7491094	14400555	−6993606
700	−72503.38	−513144	−3064817	1601468	2049060
800	70919.84	586343.25	787740.75	−886619.25	−558243.75
900	436916.5	1537.75	635731.38	−511936	−562132
1000	−2682893.03	42945.63	−901415.25	−1409781.22	4951288.04
1100	−1244105	440503.5	−11654959.5	15728878	−3270228.12

The eigenvalues of A are $\lambda = [\,233.3505,\ 27.1460,\ -8.3339,\ -43.2339,\ -62.9287\,]$.

The above example shows that in general, the identity matrix can not be used for the starting guess matrix of eigenvectors for a nondefective matrix with distinct eigenvalues.

If by another method, a matrix of eigenvectors of a matrix A can be determined, then this method can be used to find the eigenvalues, and corresponding eigenvectors of a slightly perturbed matrix $A + \epsilon E$, where E is an arbitrary matrix, and ϵ is a small scalar. For example, first reduce the matrix A to an upper Hessenberg form H by an orthogonal matrix Q, and then apply the QR method with double implicit shift to H to obtain a quasi-upper triangular matrix. Next solve $(H - \mu I)\, v = 0$ to find corresponding eigenvector v of the eigenvalue μ. Then $x = Qv$ is the eigenvector of A corresponding to the eigenvalue μ. Now these eigenvectors can be used as approximate eigenvectors in the diagonalization method to find the eigenvalues, and corresponding eigenvectors of the perturbed matrix $A + \epsilon E$.

Example 1.0.3 We reduce the matrix A in Example 1.0.2, first to an upper Hessenberg form, and then apply the QR method with double implicit shift with $tol = 10^{-6}$. After 6 iterations, we obtain a real Schur decomposition $S^T AS = U$, and the eigenvalues of

U are $\lambda = [233.3505,\ 27.1460,\ -8.3339,\ -43.2339,\ -62.9287]$. To find an eigenvector x corresponding to the eigenvalue μ_i, let $B = A - \mu_i I$. Then solve $Bx = 0$ for x. Here is the matrix of eigenvectors:

$$X = \begin{bmatrix} 0.4674 & 0.5909 & 0.3002 & -0.7147 & 0.2386 \\ 0.4487 & -0.6005 & 0.0657 & -0.1294 & 0.3043 \\ 0.4730 & 0.2783 & -0.4712 & 0.3346 & 0.5919 \\ 0.4986 & 0.1176 & -0.6361 & 0.0707 & -0.4198 \\ 0.3284 & -0.4460 & 0.5280 & 0.5962 & -0.5691 \end{bmatrix}.$$

Next consider the following matrices A_1, and A_2, which we get by perturbing the elements of A:

$$A_1 = \begin{bmatrix} 55.056 & 7.065 & 61.012 & 51.021 & 79.023 \\ 47.032 & 52.034 & 21.045 & 61.067 & 58.076 \\ 74.078 & 33.087 & 4.013 & 79.031 & 60.024 \\ 68.042 & 56.035 & 59.053 & 29.046 & 52.057 \\ 42.075 & 73.068 & 26.086 & 20.079 & 6.097 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} 55.56 & 7.65 & 61.12 & 51.21 & 79.23 \\ 47.32 & 52.34 & 21.45 & 61.67 & 58.76 \\ 74.78 & 33.87 & 4.13 & 79.31 & 60.24 \\ 68.42 & 56.35 & 59.53 & 29.46 & 52.57 \\ 42.75 & 73.68 & 26.86 & 20.79 & 6.97 \end{bmatrix}.$$

If we use Algorithm 1.0.1 to A_1, and A_2 with $X_0 = X$, and $tol = 10^{-6}$, then after 3 iterations we obtain diagonalizations $Y^{-1}A_1Y = D_1$, and $Z^{-1}A_2Z = D_2$, whereas if we apply the QR method with double implicit shift to S^TA_1S, and S^TA_2S, then after 4 iterations we obtain real Schur decompositions $P^T(S^TA_1S)P = \Omega$, and $Q^T(S^TA_1S)Q = \Gamma$.

The derivation of the results in (1.8), and (1.9) can be found in many Numerical Linear Algebra texts. The derivation of the formulas, which are used in Algorithm 1.11 can be found in [Hag88]. The result in (1.9) holds only when the eigenvalues of A are all distinct. So a natural question is to ask whether we can derive equivalent results for matrices with multiple eigenvalues.

In Section 2.1, we will show how Algorithm 1.11 can be generalized to diagonalize a nondefective matrix with multiple eigenvalues. In Section 2.4, we will show that the generalized algorithm converges locally quadratically. A Differential Equation Approach to Eigencomputations is introduced in Section 2.2 to create an iterative method to find the eigenvalues, and corresponding eigenvectors of a nondefective matrix A. The differential equations governing the block matrices have the form:

$$\dot{\Lambda}(t) = -\Lambda(t) + \text{Diag}\left(X(t)^{-1}AX(t)\right), \quad \text{and } \dot{X}(t) = X(t)F\left(X(t), \Lambda(t)\right),$$

where Diag(M) is a block diagonal matrix formed from the diagonal blocks of the block matrix M, $F_{jj}(X(t), \Lambda(t))$ is a square zero matrix, and $F_{ij}(X(t), \Lambda(t))$ for $i \neq j$ is the solution B to the matrix equation $B\Lambda_j(t) - \Lambda_i(t)B = Y_i(t)^T AX_j(t)$. Here $Y_i(t)^T$ denotes the ith block row of $X(t)^{-1}$. The Euler approximation to the differential equations can be expressed as:

$$\Lambda_{n+1}(t) = \Lambda_n + \Delta t\left(\text{Diag}\left(X_n^{-1}AX_n\right) - \Lambda_n\right), \text{ and } X_{n+1}(t) = X_n + \Delta t X_n F\left(X_n, \Lambda_n\right). \tag{1.12}$$

In Section 2.3, we modify (1.12) by replacing the fixed stepsize Δt in each iteration by a variable stepsize s. Armijo's rule from optimization theory is used to find a stepsize s, for which $||A - X_{n+1}(s)\Lambda_{n+1}(s)X_{n+1}^{-1}(s)||_F \leq ||A - X_n\Lambda_n X_n^{-1}||_F$ holds. Example 2.3.2 shows that, when a good approximation to the matrix of eigenvectors is not known, the algorithm in Section 2.3 may not converge. In Section 2.5, we discuss the factors sensitivity in the Schur decomposition relative to perturbations in the coefficient matrix to compute a block Schur decomposition of a matrix. In Section 2.6, we present an algorithm to compute a block Schur decomposition of a matrix. This algorithm is based on sensitivity results for the factors in the Schur decomposition relative to perturbations in the coefficient matrix, and Armijo's rule

from optimization theory. The starting eigenvalues associated with the Schur decomposition are uniformly distributed points on a circle in the complex plane, where the circle includes all Gerschgorin disks for the matrix, and the starting guess unitary factor is the identity matrix.

In Chapter 3, we discuss how to exploit the structure of a matrix. Since symmetric matrices are diagonalizable, we modify the diagonalization method, and the block diagonalization method to exploit the matrix symmetry and reduce the given matrix into a diagonal matrix with eigenvalues along the diagonal.

CHAPTER 2
EIGENVALUES OF UNSYMMETRIC MATRICES

2.1 Block Diagonalization of a Matrix

Let A be an $n \times n$ nondefective matrix. Our aim is to find an iterative method to compute a block diagonalization of the form $A = X\Lambda X^{-1}$, where $\Lambda = \text{diag}(\Lambda_1, \Lambda_2, \ldots, \Lambda_t)$ is a block diagonal matrix, and $X = [X_1, X_2, \ldots, X_t]$ is a compatible block column matrix such that $AX_j = X_j\Lambda_j$. Here $\Lambda_j = \text{diag}(\lambda_j, \lambda_j, \ldots, \lambda_j)$ is an $m_j \times m_j$ scalar matrix. It can always be arranged so that the eigenvalues of Λ_i and Λ_j for $i \neq j$ are distinct. We will use a sensitivity result for eigenvalues and eigenvectors to block diagonalize a matrix. In the derivation of necessary equations to block diagonalize a matrix, we need the continuity of eigenvalues, and the perturbation theory based on Gerschgorin's theorem.

Recall that the coefficients of the characteristic polynomial of a matrix B are continuous functions of the elements of B [Wil65, page 66], and the zeros of a polynomial depend continuously on its coefficients. Proof is in [Hen74, page 281]. So the eigenvalues of B depend continuously on the elements of B. Also, Rouché's theorem states that if f and g are analytic in a neighborhood of a closed disk $D = \{z : |z - a| \leq R\}$ centered at a, f has no zeros on $\partial D = \{z : |z - a| = R\}$, and $|g(z)| < |f(z)|$ holds for $z \in \partial D$, then $f(z)$, and $f(z) + g(z)$ have the same number of zeros in D. Detail is in [Hen74, page 280].

Theorem 2.1.1 *Let λ be an eigenvalue of A of algebraic multiplicity k. Then for any norm $||\cdot||$, and all sufficiently small $\eta > 0$, there is a $\delta > 0$ such that if $||E|| < \delta$,*

then the disk $D = \{z \in \mathbb{C} : |z - \lambda| \leq \eta\}$ *centered at* λ *contains precisely* k *eigenvalues of* $A + E$.

Proof of Theorem 2.1.1 Proof is given in [Ste90, page 167].

Next we will discuss the effect of perturbations in the elements of A on the eigenvalues, and the eigenvectors in its diagonalization, which is based on Gerschgorin's theorem. Let A be an $n \times n$ nondefective matrix. Suppose $X^{-1}AX = \Lambda$ is a block diagonalization of A, where Λ is a block diagonal matrix. If Λ_j denotes the jth diagonal block of Λ, then we have $AX_j = X_j\Lambda_j$, where X_j is the jth block column of X. Given an arbitrary matrix E, and for a sufficiently small ϵ, we will show in the next theorem that there exist continuously differentiable functions $\Lambda_j(\epsilon)$, and $X_j(\epsilon)$ with $\Lambda_j(0) = \Lambda_j$, and $X_j(0) = X_j$ such that $(A + \epsilon E)X_j(\epsilon) = X_j(\epsilon)\Lambda_j(\epsilon)$.

Theorem 2.1.2 *Let* A *be an* $n \times n$ *nondefective matrix. Suppose* $X^{-1}AX = \Lambda$ *is a block diagonalization of* A, *and if* Λ_j *is the* j*th diagonal block of* Λ, *then* $AX_j = X_j\Lambda_j$, *where* X_j *is the* j*th block column of* X. *Suppose* Λ_i, *and* Λ_j *for* $i \neq j$ *have distinct eigenvalues. Then given an arbitrary* $n \times n$ *matrix* E, *and for a sufficiently small* ϵ, *there exist continuously differentiable functions* $\Lambda_j(\epsilon)$, *and* $X_j(\epsilon)$ *with* $\Lambda_j(0) = \Lambda_j$, *and* $X_j(0) = X_j$ *such that the following matrix equation holds:*

$$(A + \epsilon E)X_j(\epsilon) = X_j(\epsilon)\Lambda_j(\epsilon). \tag{2.1}$$

Proof of Theorem 2.1.2 Let $X^{-1}AX = \text{diag}(\Lambda_1, \Lambda_2, \ldots, \Lambda_t)$ be a block diagonalization of A, where $\Lambda_j = \text{diag}(\lambda_j, \ldots, \lambda_j)$ is a scalar matrix, and let $m_j = \dim(\Lambda_j)$. Suppose $m_1 > 1$. Let E be an $n \times n$ matrix, and ϵ be a small number. Consider the

matrix $D = X^{-1}(A + \epsilon E)X$. The form of this matrix is illustrated below for $t = 5$:

$$D = \begin{bmatrix} \Lambda_1 & & & & \\ & \Lambda_2 & & & \\ & & \Lambda_3 & & \\ & & & \Lambda_4 & \\ & & & & \Lambda_5 \end{bmatrix} + \epsilon \begin{bmatrix} F_{11} & F_{12} & F_{13} & F_{14} & F_{15} \\ F_{21} & F_{22} & F_{23} & F_{24} & F_{25} \\ F_{31} & F_{32} & F_{33} & F_{34} & F_{35} \\ F_{41} & F_{42} & F_{43} & F_{44} & F_{45} \\ F_{51} & F_{52} & F_{53} & F_{54} & F_{55} \end{bmatrix},$$

where $F_{ij} = Y_i^T E X_j$, Y_i^T is the ith block row of X^{-1}. Let $B = \text{diag}(\, B_1, B_2, \ldots, B_t\,)$, where $B_1 = \text{diag}\left(\frac{\epsilon}{p}, \ldots, \frac{\epsilon}{p}\right)$, $B_i = I$ for $i = 2, \ldots, t$, and $\dim(B_j) = m_j$, $j = 1, \ldots, t$. Let $C = BDB^{-1}$, then

$$C = \begin{bmatrix} \Lambda_1 & & & & \\ & \Lambda_2 & & & \\ & & \Lambda_3 & & \\ & & & \Lambda_4 & \\ & & & & \Lambda_5 \end{bmatrix} + \begin{bmatrix} \epsilon F_{11} & \frac{\epsilon^2}{p} F_{12} & \frac{\epsilon^2}{p} F_{13} & \frac{\epsilon^2}{p} F_{14} & \frac{\epsilon^2}{p} F_{15} \\ pF_{21} & \epsilon F_{22} & \epsilon F_{23} & \epsilon F_{24} & \epsilon F_{25} \\ pF_{31} & \epsilon F_{32} & \epsilon F_{33} & \epsilon F_{34} & \epsilon F_{35} \\ pF_{41} & \epsilon F_{42} & \epsilon F_{43} & \epsilon F_{44} & \epsilon F_{45} \\ pF_{51} & \epsilon F_{52} & \epsilon F_{53} & \epsilon F_{54} & \epsilon F_{55} \end{bmatrix}.$$

Next, let $U^{-1}F_{11}U = \Sigma$ be a diagonalization of F_{11}, where $\Sigma = \text{diag}(\, \sigma_1, \sigma_2, \ldots, \sigma_{m_1}\,)$ with some of σ_i's may be identical. Let $W = \text{diag}(\, U, B_2, \ldots, B_t\,)$. Then

$$W^{-1}CW = \begin{bmatrix} \Lambda_1 & & & & \\ & \Lambda_2 & & & \\ & & \Lambda_3 & & \\ & & & \Lambda_4 & \\ & & & & \Lambda_5 \end{bmatrix} +$$

$$\begin{bmatrix} \epsilon\Sigma & \frac{\epsilon^2}{p} U^{-1}F_{12} & \frac{\epsilon^2}{p} U^{-1}F_{13} & \frac{\epsilon^2}{p} U^{-1}F_{14} & \frac{\epsilon^2}{p} U^{-1}F_{15} \\ pF_{21}U & \epsilon F_{22} & \epsilon F_{23} & \epsilon F_{24} & \epsilon F_{25} \\ pF_{31}U & \epsilon F_{32} & \epsilon F_{33} & \epsilon F_{34} & \epsilon F_{35} \\ pF_{41}U & \epsilon F_{42} & \epsilon F_{43} & \epsilon F_{44} & \epsilon F_{45} \\ pF_{51}U & \epsilon F_{52} & \epsilon F_{53} & \epsilon F_{54} & \epsilon F_{55} \end{bmatrix}.$$

We may choose a value of p, which is independent of ϵ such that the first m_1 disks are disjoint from the other $n - m_1$ disks. So for a proper value of p, m_1 eigenvalues of $W^{-1}CW$ lie in the union of the m_1 disks, which are centered at $\lambda_1 + \epsilon\sigma_j$, $j = 1, \ldots, m_1$. The radius of each disk is of order ϵ^2, and the perturbation in each eigenvalue λ_1 is of order ϵ as $\epsilon \to 0$.

Next we discuss the effect of perturbations in the elements of A on the eigenvectors in its block diagonalization. As earlier, let $X^{-1}AX = \text{diag}(\Lambda_1, \Lambda_2, \ldots, \Lambda_t)$ be a block diagonalization of A, where $\Lambda_j = \text{diag}(\lambda_j, \ldots, \lambda_j)$ is a scalar matrix, and $m_j = \dim(\Lambda_j)$. Clearly the columns $x_1, x_2, \ldots, x_n$ of X form a complete set of eigenvectors. We denote eigenvectors of $A + \epsilon E$ by $x_1(\epsilon), x_2(\epsilon), \ldots, x_n(\epsilon)$, and eigenvectors of $D = X^{-1}(A + \epsilon E)X$ by $u_1(\epsilon), u_2(\epsilon), \ldots, u_n(\epsilon)$ such that $x_i(\epsilon) = Xu_i(\epsilon)$.

To illustrate, we consider a matrix D with $t = n-2$ such that $\Lambda_1 = \text{diag}(\lambda_1, \lambda_1, \lambda_1)$, $\Lambda_j = \lambda_{j+2}$, $j = 2, \ldots, n-2$:

$$D = \begin{bmatrix} \lambda_1 & & & & & \\ & \lambda_1 & & & & \\ & & \lambda_1 & & & \\ & & & \lambda_4 & & \\ & & & & \ddots & \\ & & & & & \lambda_n \end{bmatrix} + \epsilon \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} & \cdots & f_{1n} \\ f_{21} & f_{22} & f_{23} & f_{24} & \cdots & f_{2n} \\ f_{31} & f_{32} & f_{33} & f_{34} & \cdots & f_{3n} \\ f_{41} & f_{42} & f_{43} & f_{44} & \cdots & f_{4n} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ f_{n1} & f_{n2} & f_{n3} & f_{n4} & \cdots & f_{nn} \end{bmatrix},$$

where $f_{ij} = y_i^T E x_j$, and y_i^T is the ith row of X^{-1}. First consider the simple eigenvalue λ_4. Since $(\lambda_4(\epsilon), u_4(\epsilon))$ is an eigenpair of D, and $\lambda_4(0) = \lambda_4$, so we must have $u_4(0) = e_4$, which is the fourth unit vector. Let $u_4(\epsilon)$ be normalized so that its largest element is 1. For sufficiently small ϵ, claim that the fourth element of $u_4(\epsilon)$ is 1. If not, let $u_{i4}(\epsilon) = 1$ for some i, $i \neq 4$. Since $Du_4(\epsilon) = \lambda_4(\epsilon)u_4(\epsilon)$, so equating the ith component on both sides, we get

$$\lambda_4(\epsilon)u_{i4}(\epsilon) = \lambda_i u_{i4}(\epsilon) + \epsilon \sum_{j=1}^{n} f_{ij}u_{j4}(\epsilon), \tag{2.2}$$

that is

$$\lambda_4(\epsilon) - \lambda_i = \epsilon \sum_{j=1}^{n} f_{ij}u_{j4}(\epsilon).$$

Now letting $\epsilon \to 0$, we find $\lambda_4 - \lambda_i = 0$, $i \neq 4$, a contradiction. So for sufficiently small ϵ, $u_{44}(\epsilon) = 1$. Next we will show that the remaining components of $u_4(\epsilon)$ are

all less than $M\epsilon$ as $\epsilon \to 0$ for some $M > 0$. Since $|u_{j4}(\epsilon)| \leq 1$, so from (2.2), we get

$$|\lambda_4(\epsilon) - \lambda_i||u_{i4}(\epsilon)| \leq \epsilon \sum_{j=1}^{n} |f_{ij}|.$$

Let $|\lambda_4(\epsilon) - \lambda_i| \geq \frac{1}{2}|\lambda_4 - \lambda_i|$ for $i \neq 4,\ i = 1, \ldots, n$, then $|u_{i4}(\epsilon)| \leq M\epsilon$, where

$$M = \max_{i \neq 4} \left\{ \frac{2\sum_{j=1}^{n} |f_{ij}|}{|\lambda_4 - \lambda_i|} \right\}.$$

Thus $u_{i4}(\epsilon)$ are of order ϵ for $i \neq 4,\ i = 1, \ldots, n$. Again from (2.2), we obtain

$$(\lambda_4(\epsilon) - \lambda_i)\, u_{i4}(\epsilon) = \epsilon f_{i4} + \epsilon \sum_{\substack{j=1 \\ j \neq 4}}^{n} f_{ij} u_{j4}(\epsilon).$$

The second term on the right is of order ϵ^2, so for $i \neq 4,\ i = 1, \ldots, n$, we have

$$\left| u_{i4}(\epsilon) - \frac{\epsilon f_{i4}}{\lambda_4 - \lambda_i} \right| = O(\epsilon^2).$$

Now we turn to eigenvectors corresponding to the multiple eigenvalue λ_1. Suppose $u_1(\epsilon)$ is a normalized eigenvector of D corresponding to the eigenvalue $\lambda_1(\epsilon)$. Using a similar argument as in the case of the simple eigenvalue $\lambda_4(\epsilon)$, we can show that the largest element of $u_1(\epsilon)$ can not be $u_{i1}(\epsilon)$, $i = 4, \ldots, n$. In fact, these elements are of order ϵ. Therefore the normalized $u_1(\epsilon)$ must have one of the following forms:

$$\begin{bmatrix} 1 \\ p_1(\epsilon) \\ p_2(\epsilon) \\ u_{41}(\epsilon) \\ \vdots \\ u_{n1}(\epsilon) \end{bmatrix}, \text{ or } \begin{bmatrix} p_1(\epsilon) \\ 1 \\ p_2(\epsilon) \\ u_{41}(\epsilon) \\ \vdots \\ u_{n1}(\epsilon) \end{bmatrix}, \text{ or } \begin{bmatrix} p_1(\epsilon) \\ p_2(\epsilon) \\ 1 \\ u_{41}(\epsilon) \\ \vdots \\ u_{n1}(\epsilon) \end{bmatrix},$$

where $|p_i(\epsilon)| \leq 1,\ i = 1, 2$. Using the same technique, as in the case of $u_1(\epsilon)$, we can show that $u_2(\epsilon)$, and $u_3(\epsilon)$ also must have one of the above forms. Since the dimension of the diagonal block Λ_1 is three, and Λ_1 is a scalar matrix, so we have

three degrees of freedom to normalize $u_1(\epsilon)$, two degrees of freedom to normalize $u_2(\epsilon)$, and one degree of freedom to normalize $u_3(\epsilon)$. Hence we take

$$u_1(\epsilon) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ u_{41}(\epsilon) \\ \vdots \\ u_{n1}(\epsilon) \end{bmatrix}, \quad u_2(\epsilon) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ u_{42}(\epsilon) \\ \vdots \\ u_{n2}(\epsilon) \end{bmatrix}, \quad \text{and} \quad u_3(\epsilon) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ u_{43}(\epsilon) \\ \vdots \\ u_{n3}(\epsilon) \end{bmatrix}.$$

As in the case of the simple eigenvalue $\lambda_4(\epsilon)$, we can show that

$$\left| u_{ij}(\epsilon) - \frac{\epsilon f_{ij}}{\lambda_j - \lambda_i} \right| = O(\epsilon^2), \ i = 4, \ldots, n, \ j = 1, 2, 3,$$

where $\lambda_1 = \lambda_2 = \lambda_3$. Hence $\Lambda_j(\epsilon)$, and $X_j(\epsilon)$ are continuously differentiable functions of ϵ. This completes the proof of the theorem. □

Now we are ready to derive the necessary equations to obtain a block diagonalization $A = X\Lambda X^{-1}$ of a nondefective matrix A. Let Λ_j be the jth block diagonal element of Λ, and X_j be the corresponding block column vector of X. Given an $n \times n$ matrix E, and for a sufficiently small $\epsilon \in \mathbb{R}$, let $\Lambda_j(\epsilon)$, and $X_j(\epsilon)$ be differentiable functions of ϵ such that $\Lambda_j(0) = \Lambda_j$, and $X_j(0) = X_j$. Differentiating (2.1) with respect to ϵ and then putting $\epsilon = 0$, we obtain

$$AX_j'(0) + EX_j = X_j'(0)\Lambda_j + X_j\Lambda_j'(0). \tag{2.3}$$

Since the columns of X are linearly independent, so we can express $X_j(\epsilon)$ in the following way:

$$X_j(\epsilon) = \sum_{i=1}^{t} X_i B_{ij}(\epsilon), \tag{2.4}$$

where $B_{ij}(\epsilon)$ is an $m_i \times m_j$ matrix. We normalize (2.4) by taking $B_{jj}(\epsilon) = I$. Differentiating (2.4) with respect to ϵ and setting $\epsilon = 0$, we have

$$X_j'(0) = \sum_{\substack{i=1 \\ i \neq j}}^{t} X_i B_{ij}'(0). \tag{2.5}$$

Substituting (2.5) into (2.3), we obtain

$$A\sum_{\substack{i=1\\ i\neq j}}^{t} X_i B'_{ij}(0) + EX_j = \sum_{\substack{i=1\\ i\neq j}}^{t} X_i B'_{ij}(0)\Lambda_j + X_j\Lambda'_j(0)$$

$$\sum_{\substack{i=1\\ i\neq j}}^{t} X_i\Lambda_i B'_{ij}(0) + EX_j = \sum_{\substack{i=1\\ i\neq j}}^{t} X_i B'_{ij}(0)\Lambda_j + X_j\Lambda'_j(0)$$

$$EX_j = \sum_{\substack{i=1\\ i\neq j}}^{t} X_i \left(B'_{ij}(0)\Lambda_j - \Lambda_i B'_{ij}(0)\right) + X_j\Lambda'_j(0). \tag{2.6}$$

Since

$$Y_i^T X_k = \begin{cases} 0 & \text{for } i \neq k, \\ I & \text{for } i = k, \end{cases} \tag{2.7}$$

where Y_i^T is the ith block row of X^{-1}. Premultiplying (2.6) by Y_j^T, and Y_i^T respectively, yields

$$\Lambda'_j(0) = Y_j^T EX_j, \tag{2.8}$$

$$B'_{ij}(0)\Lambda_j - \Lambda_i B'_{ij}(0) = Y_i^T EX_j. \tag{2.9}$$

Equation (2.9) is a linear equation in the unknown $B'_{ij}(0)$. Here we mention two well-known methods to find $B'_{ij}(0)$. To derive these two methods, we need the following two theorems.

Theorem 2.1.3 [Hag88, page 317] *If $B \in \mathbb{C}^{p\times p}$, and $C \in \mathbb{C}^{r\times r}$ are nondefective matrices, then the solution Z to the matrix equation*

$$ZB - CZ = R \tag{2.10}$$

is given by $Z = VWU^{-1}$, where $B = U\Sigma U^{-1}$, and $C = V\Omega V^{-1}$ are diagonalizations of B, and C respectively, and $W = (w_{ij})$ is a $r \times p$ matrix with $w_{ij} = \dfrac{(V^{-1}RU)_{ij}}{\sigma_j - \omega_i}$.

Proof of Theorem 2.1.3 Let $B = U\Sigma U^{-1}$, and $C = V\Omega V^{-1}$ be diagonalizations of B, and C respectively. Then from (2.10), we get

$$\begin{aligned} ZU\Sigma U^{-1} - V\Omega V^{-1}Z &= R \\ V^{-1}ZU\Sigma - \Omega V^{-1}ZU &= V^{-1}RU. \end{aligned}$$

Let $W = V^{-1}ZU$, and $D = V^{-1}RU$, then the last equation reduces to $W\Sigma - \Omega W = D$. Comparing the (i,j) entry on both sides of the preceding relation, we have

$$w_{ij}(\sigma_j - \omega_i) = d_{ij}, \quad \text{or} \quad w_{ij} = \frac{(V^{-1}RU)_{ij}}{\sigma_j - \omega_i}, \quad i = 1,2,\ldots,r, \quad j = 1,2,\ldots,p,$$

and with this W, $Z = VWU^{-1}$. □.

Theorem 2.1.4 *If $A \in \mathbb{R}^{p\times p}$, and $B \in \mathbb{R}^{r\times r}$, then the solution Z to the matrix equation*

$$AZ - ZB = C \tag{2.11}$$

is given by $Z = UPV^T$, where $A = URU^T$, and $B = VSV^T$ are real Schur decompositions of A, and B respectively, and P is the solution to the **Sylvester equation**

$$RP - PS = U^TCV.$$

Proof of Theorem 2.1.4 Let $A = URU^T$, and $B = VSV^T$ be real Schur decompositions of A, and B respectively. Then (2.11) becomes

$$\begin{aligned} URU^TZ - ZVSV^T &= C \\ RU^TZV - U^TZVS &= U^TCV. \end{aligned}$$

Let $P = U^TZV$, and $D = U^TCV$. Then the last equation reduces to $RP - PS = D$, which is a *Sylvester equation*. For the detailed solution of a Sylvester equation readers are referred to [Gol90, page 387]. □

Next we will show how to use the above two theorems to derive two methods to find the unknown $B'_{ij}(0)$ in (2.9).

Method 1 (Theorem 2.1.3): Let $P = B'_{ij}(0)$, and $R = Y_i^TEX_j$, then (2.9) reduces to

$$P\Lambda_j - \Lambda_i P = R. \tag{2.12}$$

Solving (2.12) for P, we get $P = VWU^{-1}$, where $\Lambda_j = U\Sigma U^{-1}$, and $\Lambda_i = V\Omega V^{-1}$ are diagonalizations of Λ_j, and Λ_i respectively, and $W = (w_{kl})$ with $w_{kl} = \dfrac{(V^{-1}RU)_{kl}}{\sigma_l - \omega_k}$.

Method 2 (Theorem 2.1.4): Let $P = B'_{ij}(0)$, and $C = Y_i^T E X_j$, then (2.9) becomes

$$P\Lambda_j - \Lambda_i P = C. \tag{2.13}$$

Solving (2.13) for P, we obtain $P = VZU^T$, where $\Lambda_j = URU^T$, and $\Lambda_i = VSV^T$ are real Schur decompositions of Λ_j, and Λ_i respectively, and Z is the solution to the *Sylvester equation* $SZ - ZR = W$ with $W = -V^T CU$.

Combining the analysis of the effect of perturbations in the elements of a nondefective matrix A on the eigenvalues, and corresponding eigenvectors in its diagonalization, and the results from (2.5), (2.8), and (2.9) we get the following Taylor Expansions of $\Lambda_j(\epsilon)$, and $X_j(\epsilon)$:

Theorem 2.1.5 *Let A be a nondefective matrix, $\Lambda_j = diag(\lambda_j, \ldots, \lambda_j)$ be a scalar matrix, where λ_j is an eigenvalue of A of multiplicity $dim(\Lambda_j)$; let X_j be the matrix of linearly independent eigenvectors of A corresponding to the eigenvalue λ_j such that $AX_j = X_j\Lambda_j$. Given an arbitrary matrix E, and for a sufficiently small ϵ, let $\Lambda_j(\epsilon)$, and $X_j(\epsilon)$ be continuously differentiable functions of ϵ such that $(A + \epsilon E)X_j(\epsilon) = X_j(\epsilon)\Lambda_j(\epsilon)$, $\Lambda_j(0) = \Lambda_j$, and $X_j(0) = X_j$. Then*

$$\Lambda_j(\epsilon) = \Lambda_j + \epsilon Y_j^T E X_j + O(\epsilon^2),$$

$$X_j(\epsilon) = X_j + \epsilon \sum_{\substack{i=1 \\ i \neq j}}^{t} X_i P_{ij} + O(\epsilon^2),$$

where $P = P_{ij}$ is the solution to the matrix equation $P\Lambda_j - \Lambda_i P = Y_i^T E X_j$.

We use Theorem 2.1.5 to develop an algorithm in the following way. Suppose $X\Lambda X^{-1}$ is an approximate block diagonalization of a nondefective matrix B. If we identify E in Theorem 2.1.5 with $B - X\Lambda X^{-1}$, then to the first order:

$$\Lambda_j(\epsilon) \approx \Lambda_j + \epsilon Y_j^T \left(B - X\Lambda X^{-1}\right) X_j$$

$$\begin{aligned} &\approx \Lambda_j + \epsilon Y_j^T B X_j - \epsilon \Lambda_j, \\ X_j(\epsilon) &\approx X_j + \epsilon \sum_{\substack{i=1 \\ i \neq j}}^{t} X_i P_{ij}, \end{aligned}$$

where $P = P_{ij}$ is the solution to $P\Lambda_j - \Lambda_i P = Y_i^T B X_j$.

With $\epsilon = 1$, these approximations lead to the following block version of Algorithm 1.11:

$$\begin{aligned} \Lambda_j^{new} &= \left(Y_j^{old}\right)^T B X_j^{old} \qquad \text{for } j = 1:t, \\ X_j^{new} &= X_j^{old} + \sum_{\substack{i=1 \\ i \neq j}}^{t} X_i^{old} P_{ij} \quad \text{for } j = 1:t, \\ Y^{new} &= (X^{new})^{-1}, \end{aligned} \tag{2.14}$$

where $P = P_{ij}$ is the solution to $P\Lambda_j^{new} - \Lambda_i^{new} P = \left(Y_i^{old}\right)^T B X_j^{old}$.

With a matrix of approximate eigenvectors X_0, we can use the above expressions to compute the block diagonalization of a nondefective matrix A in the following way.

Algorithm 2.1.1 (Block Diagonalization) Given an $n \times n$ nondefective matrix A, a matrix of approximate eigenvectors X_0, and a tolerance *tol* greater than the unit roundoff, the following algorithm computes a block diagonalization $A = X\Lambda X^{-1}$.

Dif = 1; $X = X_0 = \left[X_1, \ X_2, \ \ldots, \ X_k \right]$; $Y = X^{-1} = [\, Y_1, Y_2, \ldots, Y_k \,]^T$

until Dif < *tol*

$C = 0$; { $C = \left[C_1, \ C_2, \ \ldots, \ C_k \right]$ is a $n \times n$ zero matrix. }

for $j = 1:k$

$\Lambda_j = (Y_j)^T A X_j$

end

for $j = 1:k$

for $i = 1:k$

if $i \neq j$

Solve $Z\Lambda_j - \Lambda_i Z = Y_i^T A X_j$ for Z.

$C_j = C_j + Z$

end

end

$X_j = X_j + C_j$

$x_l^j = x_l^j / ||x_l^j||$ for $l = 1 : m_j$.

{ $X_j = \left[x_1^j, x_2^j, \ldots, x_{m_j}^j \right]$ is column partitioning. }

end

$Y = X^{-1}$; Dif $= ||C||_F$

end

Above, we normalize X_j to control the growth of X.

The main problem in this method is how to choose a matrix of approximate eigenvectors for the starting guess. If by another method, a matrix of eigenvectors of a matrix A can be determined, then this method can be applied with X as the starting matrix of eigenvectors to find the eigenvalues, and corresponding eigenvectors of a slightly perturbed matrix $A + \epsilon E$, where E is an arbitrary matrix, and ϵ is a small number. For example, first reduce the matrix A to an upper Hessenberg form H by an orthogonal matrix Q, and then apply the QR method with double implicit shift to H to obtain a real Schur decomposition $S^T H S = U$. To obtain an eigenvector y corresponding to the eigenvalue μ_i, let $B = H - \mu_i I$. Next solve $By = 0$ for y in the following way. Put $y(i) = 1$, and then solve $[B(:, 1 : i-1)\ B(:, i+1 : n)]z = -B(:, i)$ by using the QR factorization technique to solve an overdetermined system of equations. Put $y(1 : i-1) = z(1 : i-1)$, and $y(i+1 : n) = z(i : n-1)$. Then $x = Qy$ is the eigenvector of A corresponding to the eigenvalue μ_i. Now these eigenvectors can be used in the diagonalization method to find the eigenvalues, and corresponding eigenvectors of the perturbed matrix $A + \epsilon E$.

Example 2.1.1 Consider the matrix

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 2 & 1 \\ -1 & 3 & -1 \end{bmatrix}.$$

We reduce A, first to an upper Hessenberg form H, and then apply the QR method with double implicit shift with $tol = 10^{-6}$. After 8 iterations, we get a real Schur decomposition $S^T AS = U$. The eigenvalues of U are $\lambda = [\, 4.0394, -1.0197 + 0.4450i, -1.0197 - 0.4450i \,]$. Next, we use the technique discussed in the last paragraph to find corresponding eigenvectors. Here for each pair of complex conjugate eigenvalues, we solve the equation $By = 0$ only for one, and then take the real and the imaginary parts of y as two vectors corresponding to the complex conjugate pair of eigenvalues. The corresponding matrix is:

$$X = \begin{bmatrix} 0.5921 & 0.2979 & 0.7927 \\ 0.7387 & 0.1265 & -0.2726 \\ 0.3222 & -0.9462 & -0.5453 \end{bmatrix}$$

Next consider the following matrix A_1, which we get by perturbing the elements of A:

$$A_1 = \begin{bmatrix} 1.010 & 2.002 & 1.003 \\ 2.004 & 2.005 & 1.006 \\ -1.007 & 3.009 & -1.008 \end{bmatrix}$$

If we use Algorithm 2.1.1 to A_1 with $X_0 = X$, and $tol = 10^{-6}$, then after 2 iterations we get a block diagonalization $Y^{-1}A_1Y = D_1$, and if we apply the QR method with double implicit shift to $S^T A_1 S$, then after 2 iterations we obtain a real Schur form $Q^T(S^T A_1 S)Q = R$ as well.

Example 2.1.2 Consider the matrix

$$B = \begin{bmatrix} -10 & -9 & -3 & -6 \\ 6 & 5 & 3 & 3 \\ 6 & 6 & 2 & 3 \\ 6 & 6 & 0 & 5 \end{bmatrix}.$$

If we apply the QR method with double implicit shift with $tol = 10^{-6}$ to B, then after 1 iteration, we obtain a real Schur form $S^T BS = U$, and the eigenvalues of U are $\lambda = [-1, -1, 2, 2]$. Next, we use the technique discussed prior to Example 2.1.1 to find corresponding eigenvectors. Here is the matrix of eigenvectors:

$$X = \begin{bmatrix} 0.7559 & 0 & 0.6547 & 0 \\ -0.3780 & -0.5774 & -0.4364 & -0.4082 \\ -0.3780 & 0.5774 & -0.4364 & -0.4082 \\ -0.3780 & 0.5774 & -0.4364 & 0.8165 \end{bmatrix}.$$

Next consider the following matrix B_1, which we get by perturbing the elements of B:

$$B_1 = \begin{bmatrix} -9.9931 & -8.9947 & -2.9930 & -5.9995 \\ 6.0059 & 5.0009 & 3.0091 & 3.0074 \\ 6.0093 & 6.0065 & 2.0076 & 3.0033 \\ 6.0085 & 6.0042 & 0.0026 & 5.0063 \end{bmatrix}$$

If we apply Algorithm 2.1.1 to B_1 with $X_0 = X$, and $tol = 10^{-6}$, then after 2 iterations we obtain a block diagonalization $Y^{-1}B_1Y = D_1$, whereas if we apply the QR method with double implicit shift to $S^T B_1 S$, then after 3 iterations we obtain a real Schur form $Q^T(S^T B_1 S)Q = R$.

Example 2.1.3 Consider the matrix

$$C = \begin{bmatrix}
2 & -5 & 7 & 7 & 9 & 4 & 5 & 8 & -3 & -5 \\
5 & -9 & 7 & 4 & 7 & 9 & 8 & 3 & -5 & 7 \\
-7 & 6 & 9 & 3 & 4 & 8 & 5 & 2 & 3 & 4 \\
5 & 3 & -9 & 9 & -7 & 5 & 4 & 3 & 9 & 5 \\
6 & 7 & 7 & 10 & 9 & 2 & 3 & 4 & 5 & 6 \\
6 & 8 & -3 & 5 & 5 & 10 & 7 & 6 & 9 & 6 \\
7 & 9 & 2 & 2 & 3 & 9 & 7 & 2 & 7 & 4 \\
4 & 1 & 6 & 3 & 5 & 5 & 10 & 1 & 2 & 3 \\
9 & 4 & 1 & 9 & 4 & 1 & 9 & -9 & 2 & -7 \\
2 & 7 & 9 & 3 & 9 & 5 & 5 & 6 & 8 & 8 \\
3 & 5 & 5 & 1 & 4 & 6 & 9 & 4 & 7 & 9 \\
5 & 7 & 7 & 2 & 9 & 9 & 9 & 5 & 1 & 5 \\
2 & 10 & 2 & 6 & 6 & 0 & 0 & 8 & 7 & 3 \\
6 & 2 & 2 & -8 & 4 & 10 & 9 & 4 & 3 & 7 \\
8 & 1 & 4 & 6 & 2 & 8 & 2 & 10 & 3 & 2 \\
6 & 2 & 3 & 7 & 1 & 8 & 5 & 6 & 6 & 3 \\
5 & 3 & 6 & 5 & 4 & 3 & 4 & 4 & 4 & 5 \\
9 & 5 & 9 & 3 & 5 & 5 & 7 & 4 & 6 & 10 \\
9 & 3 & 10 & 2 & 8 & 7 & 5 & 8 & -2 & 2 \\
0 & 5 & -8 & 2 & 9 & 8 & 7 & 4 & -1 & 6
\end{bmatrix}$$

$$\begin{bmatrix}
5 & 7 & -1 & 4 & -7 & 4 & 7 & 6 & 9 & 8 \\
3 & 6 & 8 & 10 & 4 & 2 & 10 & 7 & 8 & 7 \\
2 & 5 & 9 & 9 & -6 & 9 & 5 & 5 & 3 & 10 \\
5 & 9 & -5 & 8 & 8 & 8 & 1 & -2 & 7 & 9 \\
8 & 4 & 8 & 3 & 4 & 5 & 5 & 3 & 2 & 2 \\
8 & 2 & 2 & 7 & 1 & -9 & 3 & 0 & 4 & -3 \\
4 & 5 & 1 & 6 & 8 & 6 & 10 & 6 & 1 & 10 \\
6 & 1 & 7 & 6 & 8 & 2 & 5 & 4 & 2 & -3 \\
4 & 3 & 1 & 8 & 5 & 3 & 5 & 8 & 9 & 7 \\
5 & 10 & 6 & 4 & 8 & 7 & 7 & 10 & 10 & 9 \\
7 & 8 & 7 & 7 & -2 & 9 & 5 & 5 & -7 & 7 \\
10 & 2 & 4 & 3 & 5 & 9 & 4 & 5 & 8 & 4 \\
4 & 7 & 7 & 2 & 8 & 7 & 8 & -9 & -8 & 8 \\
3 & 8 & 8 & 4 & 3 & 2 & -1 & 2 & 10 & 2 \\
1 & 2 & 6 & 7 & 8 & 7 & 8 & 7 & 6 & -6 \\
7 & 1 & 2 & 5 & 9 & 8 & 6 & 7 & 3 & 1 \\
4 & 3 & -8 & 4 & 3 & 7 & 7 & 5 & 7 & 7 \\
2 & 7 & 3 & 5 & 5 & 8 & -1 & 3 & 6 & 5 \\
7 & 7 & 4 & 6 & 8 & 10 & 7 & 3 & 8 & 2 \\
5 & -7 & 5 & 4 & 2 & 4 & 6 & 7 & 7 & 5
\end{bmatrix}$$

If we apply the QR algorithm with double implicit shift with $tol = 10^{-6}$ to C, then after 27 iterations we get a real Schur form $S^TCS = U$, and the eigenvalues of U are $\lambda = [\, 92.2615,\ -20.8808,\ 13.5193 + 10.7480i,\ 13.5193 - 10.7480i,\ 0.2459 + 15.2499i,\ 0.2459 - 15.2499i,\ 6.3218 + 13.5313,\ 6.3218 - 13.5313,\ -9.3504 + 7.8883i,\ -9.3504 - 7.8883i,\ 9.0619 + 8.7751i,\ 9.0619 - 8.7751i,\ 1.5718 + 10.4593i,\ 1.5718 - 10.4593i,\ 8.0296,\ -7.3844 + 1.4578i,\ -7.3844 - 1.4578i,\ 0.6072,\ -3.1993,\ 2.2102\ ]$. Next, we use the technique discussed prior to Example 2.1.1 to find corresponding eigenvectors. Here for each pair of complex conjugate eigenvalues, we solve the equation $Bx = 0$ only for one, and then take the real and the imaginary parts of x as two vectors corresponding to the complex conjugate pair of eigenvalues. We denote the matrix of vectors by X. Next consider the matrix

$$P = \begin{bmatrix}
7 & 7 & 9 & 2 & 7 & 5 & 9 & 6 & 9 & 8 & 3 & 7 & 5 & 2 & 1 & 3 & 9 & 7 & 5 & 10 \\
10 & 6 & 7 & 8 & 6 & 4 & 2 & 2 & 9 & 9 & 6 & 7 & 1 & 4 & 3 & 4 & 9 & 3 & 5 & 9 \\
4 & 2 & 9 & 1 & 10 & 2 & 9 & 3 & 7 & 7 & 8 & 4 & 2 & 3 & 10 & 4 & 8 & 1 & 7 & 5 \\
1 & 10 & 5 & 10 & 4 & 2 & 9 & 5 & 8 & 1 & 5 & 8 & 2 & 2 & 9 & 5 & 1 & 9 & 2 & 8 \\
3 & 9 & 7 & 2 & 9 & 3 & 4 & 6 & 8 & 3 & 0 & 2 & 6 & 8 & 6 & 3 & 4 & 9 & 3 & 3 \\
5 & 6 & 9 & 1 & 3 & 6 & 5 & 4 & 3 & 6 & 6 & 5 & 6 & 7 & 0 & 2 & 5 & 3 & 2 & 6 \\
2 & 1 & 2 & 6 & 3 & 8 & 2 & 3 & 4 & 4 & 7 & 10 & 3 & 3 & 0 & 3 & 1 & 3 & 7 & 4 \\
2 & 3 & 3 & 5 & 1 & 2 & 5 & 7 & 1 & 1 & 8 & 6 & 1 & 2 & 8 & 9 & 3 & 6 & 5 & 4 \\
4 & 2 & 5 & 9 & 7 & 2 & 9 & 8 & 5 & 6 & 6 & 8 & 1 & 5 & 6 & 4 & 4 & 6 & 0 & 4 \\
9 & 8 & 7 & 4 & 9 & 6 & 5 & 10 & 3 & 7 & 9 & 10 & 3 & 4 & 0 & 6 & 3 & 7 & 8 & 5 \\
6 & 4 & 1 & 2 & 1 & 1 & 0 & 3 & 9 & 8 & 6 & 6 & 8 & 6 & 10 & 10 & 7 & 4 & 1 & 7 \\
2 & 8 & 9 & 4 & 2 & 9 & 2 & 10 & 3 & 0 & 6 & 6 & 7 & 6 & 6 & 4 & 3 & 6 & 8 & 0 \\
7 & 1 & 0 & 5 & 8 & 7 & 5 & 2 & 5 & 6 & 5 & 4 & 0 & 4 & 3 & 1 & 8 & 9 & 2 & 7 \\
2 & 8 & 9 & 3 & 8 & 2 & 5 & 8 & 6 & 1 & 7 & 1 & 0 & 9 & 1 & 6 & 3 & 4 & 9 & 2 \\
7 & 8 & 6 & 1 & 4 & 6 & 10 & 7 & 10 & 2 & 8 & 2 & 8 & 9 & 7 & 4 & 3 & 8 & 2 & 2 \\
7 & 9 & 6 & 1 & 9 & 6 & 3 & 0 & 8 & 7 & 7 & 5 & 6 & 5 & 0 & 1 & 2 & 3 & 3 & 7 \\
1 & 3 & 9 & 0 & 6 & 3 & 3 & 5 & 2 & 2 & 2 & 1 & 9 & 2 & 4 & 4 & 0 & 1 & 6 & 5 \\
4 & 9 & 7 & 1 & 6 & 8 & 0 & 1 & 5 & 1 & 4 & 5 & 5 & 4 & 8 & 8 & 2 & 4 & 5 & 2 \\
9 & 9 & 3 & 7 & 2 & 6 & 6 & 0 & 0 & 2 & 1 & 4 & 1 & 6 & 1 & 5 & 9 & 1 & 2 & 5 \\
8 & 7 & 5 & 7 & 3 & 8 & 2 & 9 & 4 & 3 & 1 & 9 & 10 & 6 & 4 & 3 & 2 & 5 & 8 & 6
\end{bmatrix}$$

If we use Algorithm 2.1.1 to the perturbed matrix $C_1 = C + P/1000$ with $X_0 = X$, and $tol = 10^{-6}$, then after 3 iterations we obtain a block diagonalization $Y^{-1}C_1Y = D_1$,

whereas if we apply the QR method with double implicit shift to $S^T C_1 S$, then after 27 iterations we obtain a real Schur form $Q^T(S^T C_1 S)Q = R$.

2.2 A Differential Equation Approach to Eigencomputations

In the previous section, we found that the main problem in using the block diagonalization method is how to choose a matrix of approximate eigenvectors for the starting guess. Here we will study, whether the Euler method can be used to block diagonalize a nondefective matrix A with the identity matrix as the matrix of approximate eigenvectors. In numerical ordinary differential equations, one way to obtain the Euler method is to drop the 2nd order error term in the Taylor Series of the given function. In Theorem 2.1.5 dropping the 2nd order error terms, we get the expressions for $\Lambda_j(\epsilon)$, and $X_j(\epsilon)$, which can be used in Euler's method. Now we will discuss how to implement the Euler method to find the eigenvalues of a nondefective matrix A. Euler's method for the differential equation $\dot{z}(t) = f(z(t))$ has the form:

$$z_{n+1} = z_n + \Delta t f(z_n),$$

where z_n is the approximation to $z(n\Delta t)$, and Δt is the constant time step. Here z, and f are vector functions of t in general. In the block diagonalization procedure, we attempt to generate a block diagonal matrix Λ, and an associated block matrix X such that $AX = X\Lambda$. If Λ_j denotes the jth diagonal block of Λ, then we have $AX_j = X_j\Lambda_j$. The differential equation governing the block matrices has the form:

$$\dot{\Lambda}(t) = -\Lambda(t) + \mathrm{Diag}\left(X(t)^{-1}AX(t)\right), \quad \text{and} \quad \dot{X}(t) = X(t)F(X(t),\Lambda(t)),$$

where $\mathrm{Diag}(M)$ is a block diagonal matrix formed from the diagonal blocks of the block matrix M, $F_{jj}(X(t),\Lambda(t))$ is a square zero matrix, and $F_{ij}(X(t),\Lambda(t))$ for $i \neq j$ is the solution B to the matrix equation $B\Lambda_j(t) - \Lambda_i(t)B = Y_i(t)^T AX_j(t)$. Here $Y_i(t)^T$ denotes the ith block row of $X(t)^{-1}$. For convenience, we will write $\Lambda = \Lambda(t)$, and

$X = X(t)$. Typical starting guess is $\Lambda(0) = \text{Diag}(A)$, and $X(0) = I$. Hence the Euler approximation to the differential equations can be expressed as:

$$\Lambda_{n+1} = \Lambda_n + \Delta t \left(\text{Diag}\left(X_n^{-1} A X_n\right) - \Lambda_n\right), \quad \text{and} \quad X_{n+1} = X_n + \Delta t X_n F\left(X_n, \Lambda_n\right). \tag{2.15}$$

We will use the values of Λ_{n+1}, and X_{n+1} from (2.15) to create an iterative method to compute a block diagonalization of an $n \times n$ real matrix A. But first, we will discuss how to compute a block diagonalization of an upper triangular matrix. For any upper triangular matrix T, its eigenvalues are the diagonal elements. We will show how to compute a block diagonalization $Y^{-1}TY = D$ of T, where $D = \text{diag}(\, D_{11}, D_{22}, \ldots, D_{ll}\,)$ with the property that each diagonal block is upper triangular, and the eigenvalues of D_i, and D_j for $i \neq j$ are distinct. Let $Y = I_n$, and $\alpha_{ij} = |t_{ii} - t_{jj}|$. Given a tolerance tol, if $\alpha_{ij} > tol$, then define $z = -\dfrac{t_{ij}}{t_{ii} - t_{jj}}$. Let W_{ij} be equal to the identity matrix except for the (i, j) element, which is z. Then the product $W_{ij}^{-1} T W_{ij}$ is an upper triangular matrix, whose (i, j) element is zero. Next update Y by $Y^{new} = Y^{old} W_{ij}$. We use this technique to zero t_{ij}, whenever $\alpha_{ij} > tol$, $1 \leq i < j \leq n$. An efficient way to zero t_{ij}, when $\alpha_{ij} > tol$, $1 \leq i < j \leq n$, is to start from the bottom right corner of the matrix. Zero t_{ij} in the decreasing order of the row index i until $i = 1$. If $\alpha_{ij} \leq tol$ is encountered, then momentarily stop zeroing on the jth column and go to the $(j-1)$th column. Continue this process all the way to the second column. Once the second column is done, then in the increasing order of the column index j go to those columns, where some of the t_{ij} are not zeroed in the first round, even though the corresponding α_{ij} are greater than the tol. This time also zero t_{ij} in the decreasing order of the row index i until $i = 1$, whenever the corresponding $\alpha_{ij} > tol$. Then use permutations of rows and columns to obtain $D_{11}, D_{22}, \ldots, D_{ll}$. Use these permutations to Y to switch rows and columns of Y.

In fact, the above technique can be extended to a $k \times k$ quasi-upper triangular matrix $T \in \mathbb{R}^{n\times n}$. In this case W_{ij} is equal to the $k \times k$ block identity matrix except for the (i,j) block, which is Z, where Z is the solution to the matrix equation $T_{ii}Z - ZT_{jj} = -T_{ij}$. We summarize the block diagonalization of T in the following algorithm.

Algorithm 2.2.1 Given a $k \times k$ quasi-upper triangular matrix $T \in \mathbb{R}^{n\times n}$, and a coalescing tolerance *tol* greater than the square root of the unit roundoff, the following algorithm computes a block diagonalization of T.

$S = I_n$ { S is the $n \times n$ identity matrix. }
row = [ ]; col = [ ]
for $j = k : -1 : 2$
  $i = j - 1$; minim = $2tol$
  while minim $> tol \wedge i \geq 1$
    $\alpha_{ij} = \min\{\ |\sigma - \omega|\ :\ \sigma \in \lambda(T_{ii}),$ and $\omega \in \lambda(T_{jj})\ \}.$
    if $\alpha_{ij} < tol$
      row = [i row]; col = [j col]
      { row, and col are two arrays, whose elements are the row index i,
       and the column index j respectively, of an entry T_{ij} of T which will
       not be replaced by a zero matrix. }
      minim = α_{ij}
    else
      Solve $T_{ii}Z - ZT_{jj} = -T_{ij}$ for Z, and then form W_{ij}.
      $T = W_{ij}^{-1}TW_{ij}$; $S = SW_{ij}$
    end
    $i = i - 1$

end

end

m = length(row)

for $l = 1 : m$

i = row(l) $- 1$; j = col(l)

while $i \geq 1$

if $\alpha_{ij} \geq tol$

$T = W_{ij}^{-1} T W_{ij}$; $S = S W_{ij}$

end

$i = i - 1$

end

end

Let $l = k$, and suppose NT is an array of k elements whose ith element is the dimension of the ith block of T along the diagonal. For $i < j$, define α_{ij} as in the above code. If $\alpha_{ij} < tol$, then merge blocks T_{ii} and T_{jj} to form a single block using permutations of columns and rows of T. Use those column permutations to S to merge S_i and S_j. Update k^{old}, and NT^{old} to obtain k^{new}, and NT^{new}. Try the above merging technique for all possible combinations of $1 \leq i < j \leq l$.

Now we are ready to compute a block diagonalization of an $n \times n$ nondefective matrix A using the following procedure:

Algorithm 2.2.2 (Dynamical Eigencomputations) Given a nondefective matrix $A \in \mathbb{R}^{n \times n}$, a tolerance *tol* greater than the unit roundoff, a coalescing tolerance *tol1* greater than the square root of the unit roundoff, a constant stepsize Δt smaller than 1, and a starting guess diagonal matrix Λ_0, the following algorithm uses (2.15) to compute a block diagonalization $A = X \Lambda X^{-1}$.

Define $tol2 = 100 tol1$. Next we break the sketch of the algorithm into several steps.

Step 0: Take $X = I$, $\Lambda = \Lambda_0$, , $Y = X^{-1}$, $NB = n$ (the number of blocks), and $NS = [\,1, 1, \ldots, 1\,]$; an array of n elements whose ith element is the dimension of the ith diagonal block of Λ.

Step 1: Let $m = NB$. For $i < j$, define DIF $= \min\{\, |\sigma - \omega| : \sigma \in \lambda(\Lambda_j)$, and $\omega \in \lambda(\Lambda_i)\,\}$. If DIF $< tol1$, then merge blocks Λ_i and Λ_j to form a single block using permutations of columns and rows of Λ. Use those column permutations to X to merge X_i and X_j, and those row permutations to Y to merge Y_i^T and Y_j^T. Update NB^{old}, and NS^{old} to get NB^{new}, and NS^{new}. Try the above merging technique for all possible combinations of $1 \leq i < j \leq m$. Using column and row permutations, arrange blocks of Λ in the decreasing order of sizes along the diagonal. Use those column permutations to arrange block columns of X in the decreasing order of sizes, and those row permutations to arrange block rows of Y in the decreasing order of sizes.

Step 2: Construct $F(X, \Lambda)$ as follows:

$$F_{ij}(X, \Lambda) = \begin{cases} 0 & \text{for } i = j, \\ P_{ij} & \text{for } i \neq j, \end{cases}$$

where $P = P_{ij}$ is the solution to the matrix equation $P\Lambda_j - \Lambda_i P = Y_i^T A X_j$.

Step 3: Suppose $\mu = [\Lambda_{m+1}, \Lambda_{m+2}, \ldots, \Lambda_{NB}]^T$, where $0 \leq m < NB$ with $\dim(\Lambda_i) > 1$ for $i = 1, 2, \ldots, m$, and $\dim(\Lambda_i) = 1$ for $i = m+1, m+2, \ldots, NB$. Define $d = \operatorname{diag}(YAX)(t : n) - \mu$, where $t = 1 + \sum_{i=1}^{m} NS(i)$, and $\operatorname{diag}(M)(t : n)$ is a vector formed from t through n elements of the vector $\operatorname{diag}(M)$. Our aim is to find the smallest $s \in (0, 1)$ with the property $\mu_i + sd_i = \mu_j + sd_j$. To obtain this s we do the following. Let k be an array with the property that $\mu_{k(i)} \leq \mu_{k(i+1)}$. Next form the ratio:

$$r_i = \frac{\mu_{k(i+1)} - \mu_{k(i)}}{d_{k(i)} - d_{k(i+1)}},$$

and define $s = \min\{\, r_i : i = 1, 2, \ldots, n-t,\ 0 < r_i < 1 \,\}$. If $s < \Delta t$, then take a time step of size s instead of the normal time step Δt. Otherwise take the normal time step Δt.

Step 4: Update Λ, X, and Y as follows:

$$\begin{aligned} \Lambda_j^{new} &= \Lambda_j^{old} + \Delta t \left(\left(Y_j^{old} \right)^T A X_j^{old} - \Lambda_j^{old} \right), \quad j = 1, 2, \ldots, NB, \\ X^{new} &= X^{old} \left(I + \Delta t F \left(X^{old}, \Lambda^{old} \right) \right), \\ Y^{new} &= \left(X^{new} \right)^{-1}. \end{aligned}$$

Step 5: Consider the block Λ_j with $1 \le j \le m$, where m is defined as in Step 3. Let $\Lambda_j = QUQ^T$ be a Schur decomposition of Λ_j. Then $U = D + N$, where $D = \text{diag}(\, U_{11}, U_{22}, \ldots, U_{ll} \,)$, and N is the strictly upper triangular part of U. U_{ii} is either a 1×1 matrix, or a 2×2 matrix. Let NU be an array whose ith element is the dimension of the block U_{ii}. Use Algorithm 2.2.1 with *tol2* as the coalescing tolerance to reduce U to a block diagonal matrix, to update the array NU and the number of blocks l, and to get the invertible matrix S. Replace Λ_j by U, X_j by $X_j QS$, Y_j by $S^{-1}Q^T Y_j$, NB by $NB + l - 1$, and the size of the block Λ_j in NS by NU. Try the above decoupling procedure for all j such that $1 \le j \le m$.

Step 6: Compute $f = ||A - X\Lambda Y||_F$. Goto Step 1 until $f < tol$.

Example 2.2.1 If Algorithm 2.2.2 is applied to

$$A = \begin{bmatrix} 3 & -3 & 4 \\ -1 & -2 & 3 \\ 3 & 4 & 2 \end{bmatrix} \quad \text{with} \quad \Lambda_0 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 2 \end{bmatrix},$$

$tol = 0.01$, $tol1 = 10^{-4}$, and $\Delta t = 0.05$, then after 130 iterations the diagonal matrix Λ_0 converges to $\Lambda = \text{diag}(\, 6.11220, -5.34158, 2.22938 \,)$.

Example 2.2.2 Similarly, if we apply Algorithm 2.2.2 to the matrix

$$A = \begin{bmatrix} -2 & -4 & 1 \\ 2 & 4 & 1 \\ 3 & -1 & 2 \end{bmatrix} \quad \text{with} \quad \Lambda_0 = \begin{bmatrix} -2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 2 \end{bmatrix},$$

$tol = 0.01$, $tol1 = 10^{-4}$, and $\Delta t = 0.05$, then it takes 131 iterations to the diagonal matrix Λ_0 to converge to

$$D = \text{diag}\left(\begin{bmatrix} -37.00078 & 27.61726 \\ -58.11556 & 42.99974 \end{bmatrix}, -1.99896 \right).$$

The eigenvalues of D are $\lambda = [\, 2.99948 + 2.22976i,\ 2.99948 - 2.22976i,\ -1.99896 \,]$.

Example 2.2.3 If we apply Algorithm 2.2.2 to the matrix

$$A = \begin{bmatrix} 1 & 2 & -1 & 1 & 3 & -1 & -2 & -4 \\ 4 & 2 & 1 & -1 & -3 & 1 & -2 & -1 \\ -1 & -1 & 3 & 1 & 1 & -2 & -2 & 2 \\ 3 & -3 & 1 & 4 & -2 & -1 & 1 & -1 \\ 4 & 3 & -3 & -1 & -1 & 2 & 2 & 1 \\ -3 & -1 & -2 & 1 & -1 & -2 & 4 & 3 \\ -1 & -2 & -2 & 2 & 2 & 3 & -4 & 1 \\ -2 & 2 & -1 & 1 & -3 & 3 & 4 & -3 \end{bmatrix}$$

with $\Lambda_0 = \text{diag}(\, 1,\ 2,\ 3,\ 4,\ -1,\ -2,\ -4,\ -3 \,)$, $tol = 0.01$, $tol1 = 10^{-4}$, and $\Delta t = 0.01$, then it takes 802 iterations to the diagonal matrix Λ_0 to converge to

$$D = \text{diag}\left(\begin{bmatrix} -109.0896 & 62.4219 \\ -193.3755 & 110.5633 \end{bmatrix}, \begin{bmatrix} 15.3313 & -35.8551 \\ 13.3030 & -28.3276 \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} -1.2449 & -196.8223 \\ 0.1837 & 10.7697 \end{bmatrix}, 6.8342,\ -4.8365 \right).$$

The eigenvalues of D are $\lambda = [\, 0.7369 + 3.0020i,\ 0.7369 - 3.0020i,\ -6.4982 + 0.6740i,$ $-6.4982 - 0.6740i,\ 4.7624 + 0.2597i,\ 4.7624 - 0.2597i,\ 6.8342,\ -4.8365 \,]$.

This method has the following disadvantage. As the size of the given matrix A grows, we need to take a stepsize smaller than 0.1. However with a stepsize smaller than 0.1, too many iterations are required to obtain a few decimal places of accuracy.

2.3 A Differential Equation Approach to Eigencomputations with Armijo's Stepsize

In the Differential Equation Approach to Eigencomputations, we found that with a constant time step Δt, which is usually a small number, too many iterations are required to achieve a few digits accuracy in the results. So here we will modify the Differential Equation Approach to Eigencomputations by varying the time step Δt in each iteration. We plan to achieve this by using Armijo's rule from optimization theory.

Let A be an $n \times n$ real nondefective matrix. We will find an iterative method to compute a block diagonalization of the form $A = X\Lambda X^{-1}$, where $\Lambda = \text{diag}(\Lambda_1, \Lambda_2, \ldots, \Lambda_t)$ is a block diagonal matrix, and $X = [X_1, X_2, \ldots, X_t]$ is a compatible block column matrix such that $AX_j = X_j\Lambda_j$. As in Section 2.2, for the block eigenvalue problem, the differential equations that we solve are:

$$\dot{\Lambda}(t) = -\Lambda(t) + \text{Diag}\left(X(t)^{-1}AX(t)\right), \quad \text{and} \quad \dot{X}(t) = X(t)F\left(X(t), \Lambda(t)\right),$$

where $\text{Diag}(M)$ is a block diagonal matrix formed from the diagonal blocks of a block matrix M, $F_{jj}(X(t), \Lambda(t))$ is a square zero matrix, and $F_{ij}(X(t), \Lambda(t))$ for $i \neq j$ is the solution B to the matrix equation $B\Lambda_j(t) - \Lambda_i(t)B = Y_i(t)^T AX_j(t)$. Here $Y_i(t)^T$ denotes the ith block row of $X(t)^{-1}$. Hence the Euler approximation to the differential equations can be expressed as:

$$\Lambda_{n+1} = \Lambda_n + \Delta t\left(\text{Diag}\left(X_n^{-1}AX_n\right) - \Lambda_n\right), \quad \text{and} \quad X_{n+1} = X_n + \Delta t X_n F\left(X_n, \Lambda_n\right). \tag{2.16}$$

In Euler's method the normal stepsize Δt is constant. Here we will vary the stepsize Δt in each iteration. Let s be a positive parameter, and let $\Omega(s)$, and $Z(s)$ be defined by

$$\Omega(s) = \Lambda_n + s\left(\text{Diag}\left(X_n^{-1}AX_n\right) - \Lambda_n\right), \quad \text{and} \quad Z(s) = X_n + sX_nF\left(X_n, \Lambda_n\right). \tag{2.17}$$

So $\Omega(0) = \Lambda_n$, $Z(0) = X_n$, and $\Omega(\Delta t) = \Lambda_{n+1}$, $Z(\Delta t) = X_{n+1}$, the matrices generated by a Euler step. Define $f(s) = ||G(s)||_F$, where $G(s) = A - Z(s)\Omega(s)Z(s)^{-1}$. Hence $f(0) = ||A - X_n\Lambda_n X_n^{-1}||_F$, and $f(\Delta t) = ||A - X_{n+1}\Lambda_{n+1}X_{n+1}^{-1}||_F$.

If the starting guess block diagonal matrix Λ_0, and the starting guess invertible matrix X_0 are good approximations of Λ, and X in the factorization $A = X\Lambda X^{-1}$, then we must have $f(\Delta t) \leq f(0)$. Here our goal is to find an s, $0 < s \leq 1$, for which $f(s) \leq f(0)$ holds. To this end, we use Armijo's rule from optimization theory. In Armijo's rule, we determine s in the following way. Evaluate $f(s)$ at $s = 1, \frac{1}{2}, \frac{1}{4}, \ldots$, stopping when

$$f(s) \leq \left(1 - \frac{s}{2}\right) f(0).$$

Simplifying the above inequality we get

$$\frac{f(s) - f(0)}{s} \leq -\frac{1}{2} f(0).$$

It turns out ([Hag88, page 178 – 80]) that to use the above rule, we must have $f'(0) = -f(0)$. So our next aim is to determine $f'(0)$, when $f(s) = ||G(s)||_F$, and $G(s) = A - Z(s)\Omega(s)Z(s)^{-1}$. Suppressing the subscripts of Λ_n, and X_n in (2.17), we get

$$\Omega(s) = \Lambda + s\left(\text{Diag}\left(X^{-1}AX\right) - \Lambda\right), \text{ and } Z(s) = X + sXF(X, \Lambda).$$

Before we find $f'(0)$, we want to prove the following fundamental result.

Lemma 2.3.1 *Let $G(s)$ be an $n \times n$ complex matrix, whose elements are differentiable functions of s. If we define $f(s) = ||G(s)||_F$, then*

$$f(0)f'(0) = \textit{tracc}\left(G(0)^H \frac{d}{ds} G(s)|_{s=0}\right). \tag{2.18}$$

Proof of Lemma 2.3.1 Clearly

$$f(s)^2 = ||G(s)||_F^2 = \text{trace}\left(G(s)^H G(s)\right). \tag{2.19}$$

Differentiating (2.19) with respect to s, we have

$$\begin{aligned} 2f(s)f'(s) &= \text{trace}\left(\frac{d}{ds}\left(G(s)^H\right)G(s) + G(s)^H\frac{d}{ds}G(s)\right) \\ &= \text{trace}\left(\left(\frac{d}{ds}G(s)\right)^H G(s) + G(s)^H\frac{d}{ds}G(s)\right) \\ &= 2\,\text{trace}\left(G(s)^H\frac{d}{ds}G(s)\right). \end{aligned}$$

After simplification setting $s = 0$, we get

$$f(0)f'(0) = \text{trace}\left(G(0)^H\frac{d}{ds}G(s)|_{s=0}\right). \quad \square$$

Now with $\Omega(s) = \Lambda + s\,(\text{Diag}\,(X^{-1}AX) - \Lambda)$, $Z(s) = X + sXF\,(X,\Lambda)$, $G(s) = A - Z(s)\Omega(s)Z(s)^{-1}$, and $f(s) = ||G(s)||_F$ and the result of the above lemma, we are in a position to show that $f'(0) = -f(0)$.

Theorem 2.3.1 *Let $A \in \mathbb{R}^{n\times n}$ be a nondefective matrix. Suppose $A = X\Lambda X^{-1}$ is a block diagonalization of A, where $\Lambda = diag(\,\Lambda_1, \Lambda_2, \ldots, \Lambda_t\,)$ is a block diagonal matrix such that Λ_i, and Λ_j for $i \neq j$ have distinct eigenvalues, and $X = [\,X_1, X_2, \ldots, X_t\,]$ is a compatible block column matrix. If we define $\Omega(s) = \Lambda + s\,(Diag\,(X^{-1}AX) - \Lambda)$, $Z(s) = X + sXF\,(X,\Lambda)$, $G(s) = A - Z(s)\Omega(s)Z(s)^{-1}$, and $f(s) = ||G(s)||_F$, where $Diag(M)$ is a block diagonal matrix formed from the diagonal blocks of the block matrix M, $F_{jj}\,(X,\Lambda)$ is a square zero matrix, and $F_{ij}\,(X,\Lambda)$ for $i \neq j$ is the solution P to the matrix equation $P\Lambda_j - \Lambda_i P = Y_i^T AX_j$, where Y_i^T denotes the ith block row of X^{-1}, then $f'(0) = -f(0)$.*

Proof of Theorem 2.3.1 The result (2.18) of Lemma 2.3.1 gives

$$f(0)f'(0) = \text{trace}\left(G(0)^T\frac{d}{ds}G(s)|_{s=0}\right). \tag{2.20}$$

Differentiating the expression for $G(s)$ with respect to s, we have

$$\begin{aligned}\frac{d}{ds}G(s) &= -\frac{d}{ds}(Z(s))\,\Omega(s)Z(s)^{-1} - Z(s)\frac{d}{ds}(\Omega(s))\,Z(s)^{-1} - Z(s)\Omega(s)\frac{d}{ds}\left(Z(s)^{-1}\right)\\ &= -\frac{d}{ds}(Z(s))\,\Omega(s)Z(s)^{-1} - Z(s)\frac{d}{ds}(\Omega(s))\,Z(s)^{-1}\\ &\quad + Z(s)\Omega(s)Z(s)^{-1}\frac{d}{ds}(Z(s))\,Z(s)^{-1}. \qquad (2.21)\end{aligned}$$

Evaluating $\frac{d}{ds}G(s)$ at $s = 0$, we obtain

$$\begin{aligned}\frac{d}{ds}G(s)|_{s=0} &= -\frac{d}{ds}Z(s)|_{s=0}\Omega(0)Z(0)^{-1} - Z(0)\frac{d}{ds}\Omega(s)|_{s=0}Z(0)^{-1}\\ &\quad + Z(0)\Omega(0)Z(0)^{-1}\frac{d}{ds}Z(s)|_{s=0}Z(0)^{-1}. \qquad (2.22)\end{aligned}$$

Differentiating $\Omega(s)$, and $Z(s)$ with respect to s and then evaluating the derivatives at $s = 0$, we get

$$\frac{d}{ds}\Omega(s)|_{s=0} = -\Lambda + \mathrm{Diag}\left(X^{-1}AX\right), \quad \text{and} \quad \frac{d}{ds}Z(s)|_{s=0} = XF(X,\Lambda).$$

Using these values and values of $\Omega(0)$, and $Z(0)$ in (2.22), we have

$$\begin{aligned}\frac{d}{ds}G(s)|_{s=0} &= -XF(X,\Lambda)\Lambda X^{-1} - X\left(\mathrm{Diag}\left(X^{-1}AX\right) - \Lambda\right)X^{-1}\\ &\quad + X\Lambda X^{-1}\left(XF(X,\Lambda)\right)X^{-1}\\ &= -X\left(F(X,\Lambda)\Lambda - \Lambda F(X,\Lambda)\right)X^{-1}\\ &\quad - X\mathrm{Diag}\left(X^{-1}AX\right)X^{-1} + X\Lambda X^{-1}. \qquad (2.23)\end{aligned}$$

Next, let $D = F(X,\Lambda)\Lambda - \Lambda F(X,\Lambda)$. Then

$$D_{ij} = \begin{cases} 0 & \text{for } i = j,\\ F_{ij}\Lambda_j - \Lambda_i F_{ij} & \text{for } i \neq j, \end{cases} \qquad (2.24)$$

where $F_{ij} = F_{ij}(X,\Lambda)$. According to our assumption for $i \neq j$, F_{ij} is the solution P to the matrix equation $P\Lambda_j - \Lambda_i P = Y_i^T A X_j$. So $F_{ij}\Lambda_j - \Lambda_i F_{ij} = Y_i^T A X_j$. Hence after simplification (2.24) gives

$$F(X,\Lambda)\Lambda - \Lambda F(X,\Lambda) = X^{-1}AX - \mathrm{Diag}\left(X^{-1}AX\right). \qquad (2.25)$$

Using (2.25) into (2.23), we have

$$\begin{aligned}\frac{d}{ds}G(s)|_{s=0} &= -X\left(X^{-1}AX - \text{Diag}\left(X^{-1}AX\right)\right)X^{-1} \\ &\quad -X\text{Diag}\left(X^{-1}AX\right)X^{-1} + X\Lambda X^{-1} \\ &= -\left(A - X\Lambda X^{-1}\right). \qquad (2.26)\end{aligned}$$

Since $G(0) = A - Z(0)\Omega(0)Z(0)^{-1} = A - X\Lambda X^{-1}$, so (2.26) gives $\frac{d}{ds}G(s)|_{s=0} = -G(0)$, and with this value (2.20) reduces to

$$\begin{aligned}f(0)f'(0) &= \text{trace}\left(G(0)^T\left(-G(0)\right)\right) \\ &= -f(0)^2.\end{aligned}$$

Hence $f'(0) = -f(0)$, provided $f(0) \neq 0$. □.

The above theorem implies that Armijo's rule can be applied to Algorithm 2.2.2 mentioned in the Differential Equation Approach to Eigencomputations.

A modified algorithm to find the eigenvalues and corresponding eigenvectors of a nondefective matrix A can be obtained as follows:

Algorithm 2.3.1 Given a nondefective matrix $A \in \mathbb{R}^{n\times n}$, a tolerance *tol* greater than the unit roundoff, a coalescing tolerance *tol1* greater than the square root of the unit roundoff, an invertible matrix X_0, and a block diagonal matrix Λ_0, the following algorithm uses (2.15) to compute a block diagonalization $A = X\Lambda X^{-1}$.

Define $tol2 = 100tol1$. Next we break the sketch of the algorithm into several steps.

Step 0: Take $X = X_0$, $Y = X^{-1}$, $\Lambda = \Lambda_0$. Let NB denote the number of diagonal blocks of Λ, and let NS denote an array of NB elements whose ith element is the dimension of the ith diagonal block of Λ.

Step 1: Let $m = NB$. For $i < j$, define DIF $= \min\{\ |\sigma - \omega| : \sigma \in \lambda(\Lambda_j)$, and $\omega \in \lambda(\Lambda_i)\ \}$. If DIF $<$ *tol1*, then merge blocks Λ_i and Λ_j to form a single block using

permutations of columns and rows of Λ. Use those column permutations to X to merge X_i and X_j, and those row permutations to Y to merge Y_i^T and Y_j^T. Update NB^{old}, and NS^{old} to get NB^{new}, and NS^{new}. Try the above merging procedure for all possible combinations of $1 \leq i < j \leq m$. Using column and row permutations, arrange blocks of Λ in the decreasing order of sizes along the diagonal. Use those column permutations to arrange block columns of X in the decreasing order of sizes, and those row permutations to arrange block rows of Y in the decreasing order of sizes.

Step 2: Construct $F(X, \Lambda)$ in the following way.

Take $F_{jj}(X, \Lambda) = 0$, which is an $m_j \times m_j$ zero matrix, and for $i \neq j$, $F_{ij}(X, \Lambda)$ can be determined by using the following loop:

for $j = 1 : NB$

  for $i = 1 : NB$

    if $i \neq j$

      Solve $P\Lambda_j - \Lambda_i P = Y_i^T A X_j$ for P.

      $F_{ij}(X, \Lambda) = P$.

    end

  end

end

Step 3: Let $f(s) = ||G(s)||_F$, where $G(s) = A - X(s)\Lambda(s)X(s)^{-1}$, $X(s) = X(I + sF)$, and $\Lambda(s) = \Lambda + s(\mathrm{Diag}(X^{-1}AX) - \Lambda)$. Evaluate $f(s)$ at $s = 1, \frac{1}{2}, \frac{1}{4}, \cdots$, stopping when

$$f(s) \leq \left(1 - \frac{s}{2}\right) f(0). \tag{2.27}$$

Let z be the first value of s for which (2.27) is true.

Step 4: Suppose $\mu = [\Lambda_{m+1}, \Lambda_{m+2}, \ldots, \Lambda_{NB}]^T$, where $0 \leq m < NB$ with $\dim(\Lambda_i) > 1$ for $i = 1, 2, \ldots, m$, and $\dim(\Lambda_i) = 1$ for $i = m+1, m+2, \ldots, NB$. Define $d = \mathrm{diag}\,(YAX)\,(t:n) - \mu$, where $t = 1 + \sum_{i=1}^{m} NS(i)$, and $\mathrm{diag}(M)(t:n)$ is a vector formed from t through n elements of the vector $\mathrm{diag}(M)$. Our aim is to find two indices il, and iu as follows. Let k be an array with the property that $\mu_{k(i)} \leq \mu_{k(i+1)}$. Next form the ratio:

$$r_i = \frac{\mu_{k(i+1)} - \mu_{k(i)}}{d_{k(i)} - d_{k(i+1)}},$$

and let $r_{i_0} = \min\{\, r_i : i = 1, 2, \ldots, n-t,\ 0 < r_i < 1 \,\}$. Let $il = t - 1 + \min\{k(i_0), k(i_0+1)\,\}$, and $iu = t - 1 + \max\{\, k(i_0), k(i_0+1)\,\}$.

Step 5: Update Λ, X, and Y as follows:

$$\begin{aligned}
\Lambda_j^{new} &= \Lambda_j^{old} + z\left(\left(Y_j^{old}\right)^T AX_j^{old} - \Lambda_j^{old}\right), \quad j = 1, 2, \ldots, NB, \\
X^{new} &= X^{old}\left(I + zF\left(X^{old}, \Lambda^{old}\right)\right), \\
Y^{new} &= \left(X^{new}\right)^{-1}.
\end{aligned}$$

Step 6: Consider the block Λ_j with $1 \leq j \leq m$, where m is defined as in Step 4. Let $\Lambda_j = QUQ^T$ be a Schur decomposition of Λ_j. Then $U = D + N$, where $D = \mathrm{diag}(\, U_{11}, U_{22}, \ldots, U_{ll}\,)$, and N is the strictly upper triangular part of U. U_{ii} is either a 1×1 matrix, or a 2×2 matrix. Let NU be an array whose ith element is the dimension of the block U_{ii}. Use Algorithm 2.2.1 with *tol2* as the coalescing tolerance to reduce U to a block diagonal matrix, to update the array NU and the number of blocks l, and to get the invertible matrix S. Replace Λ_j by U, X_j by X_jQS, Y_j by $S^{-1}Q^TY_j$, NB by $NB + l - 1$, and the size of the block Λ_j in NS by NU. Try the above decoupling procedure for all j with $1 \leq j \leq m$.

Step 7: Using row and column permutations merge $\Lambda_{il,il}$ and $\Lambda_{iu,iu}$ to form a 2×2 diagonal block. Use those column permutations to X, and those row permutations to Y.

Step 8: Compute $f(0) = ||A - X\Lambda Y||_F$. Goto Step 1 until $f(0) < tol$.

Example 2.3.1 If we apply Algorithm 2.3.1 to

$$A = \begin{bmatrix} 0 & 1 & 1 \\ -2 & -2 & -1 \\ -2 & -1 & 0 \end{bmatrix} \quad \text{with} \quad \Lambda_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$X_0 = I$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 42 iterations the diagonal matrix Λ_0 converges to

$$D = \text{diag}\left(\begin{bmatrix} -6.01310 & 4.53255 \\ -5.98585 & 4.01310 \end{bmatrix}, 0 \right).$$

The eigenvalues of D are $\lambda = [\, -1 + 1.41421i,\ -1 - 1.41421i,\ 0\,]$.

Example 2.3.2 If Algorithm 2.3.1 is applied to

$$A = \begin{bmatrix} -2 & -1 & 0 \\ 0 & 1 & 1 \\ -2 & -2 & -1 \end{bmatrix} \quad \text{with} \quad \Lambda_0 = \begin{bmatrix} -2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix},$$

$X_0 = I$, the tolerance, and the coalescing tolerance are as in Example 2.3.1, then after 196 iterations the diagonal matrix Λ_0 becomes

$$D_{196} = \text{diag}(\, -1.04694840,\ 0.00224879,\ -0.95530039\,).$$

In the following 4 successive iterations D_{196} changes to

$$\begin{aligned} D_{197} &= \text{diag}(\, -1.04676152,\ 0.00223123,\ -0.95546971\,) \\ D_{198} &= \text{diag}(-1.04657539,\ 0.00221380,\ -0.95563841\,) \\ D_{199} &= \text{diag}(-1.04639002,\ 0.00219651,\ -0.95580649\,) \\ D_{200} &= \text{diag}(-1.04620540,\ 0.00217935,\ -0.95597395\,). \end{aligned}$$

In the above iterations $s = \dfrac{1}{2^6}$ is the Armijo's stepsize. From the diagonal matrices $D_{196}, \ldots, D_{200}$, it is clear that the first, and the third entries along the diagonal are slowly approaching each other.

Example 2.3.3 Consider the matrix

$$A = \begin{bmatrix} 5 & 4 & 9 & 3 & 3 & 7 & 1 \\ 2 & 2 & 6 & 7 & 3 & -2 & -2 \\ -5 & 6 & 8 & 1 & -1 & 4 & 3 \\ -6 & -9 & 6 & 3 & 7 & 7 & -1 \\ -4 & -3 & 4 & 4 & 2 & 1 & 6 \\ -2 & -2 & -2 & 6 & 8 & 9 & 2 \\ 7 & 4 & 3 & 5 & -3 & -5 & 3 \end{bmatrix}$$

If we apply the QR method with double implicit shift with $tol = 10^{-6}$ to A, then after 11 iterations, we obtain a real Schur form $S^T AS = U$, and the eigenvalues of U are $\lambda = [\, 10.8425 + 7.9032i,\ 10.8425 - 7.9032i,\ -2.6168 + 8.7556i,\ -2.6168 - 8.7556i,\ 3.5414 + 1.2899i,\ 3.5414 - 1.2899i,\ 8.4658\,]$. Next, we use the technique discussed prior to Example 2.1.1 to find corresponding eigenvectors. Here for each pair of complex conjugate eigenvalues, we solve the equation $Bx = 0$ only for one, and then take the real and the imaginary parts of x as two vectors corresponding to the complex conjugate pair of eigenvalues. We denote the matrix of vectors by X. Consider the following perturbed matrix A_1, which we obtain by perturbing the elements of A:

$$A_1 = \begin{bmatrix} 5.09 & 4.07 & 9.06 & 3.06 & 3.03 & 7.05 & 1.09 \\ 2.02 & 2.04 & 6.10 & 7.03 & 3.06 & -1.96 & -1.96 \\ -4.98 & 6.04 & 8.06 & 1.05 & -0.99 & 4.02 & 3.01 \\ -5.97 & -8.95 & 6.01 & 3.05 & 7.07 & 7.00 & -0.91 \\ -3.91 & -2.99 & 4.10 & 4.10 & 2.06 & 1.09 & 6.01 \\ -1.93 & -1.94 & -1.96 & 6.01 & 8.08 & 9.04 & 2.02 \\ 7.02 & 4.08 & 3.01 & 5.02 & -2.98 & -4.99 & 3.01 \end{bmatrix}$$

If we use Algorithm 2.3.1 to A_1 with $X_0 = X$, $\Lambda_0 = X^{-1}AX$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 3 iterations we get a block diagonalization $Y^{-1}A_1Y = D_1$, whereas if we apply the QR method with double implicit shift to $S^T A_1 S$, then after 9 iterations we obtain a real Schur form $Q^T(S^T A_1 S)Q = R$.

Example 2.3.4 Consider the matrix

$$A = Y\text{diag}(\ 1,\ 2,\ 1,\ 3,\ 2,\ 1,\ -1\ )Y^{-1},$$

where $Y = Q\text{diag}(\ 1,\ 2,\ 3,\ 4,\ -1,\ -2,\ -3\ )Q^T$, and Q is an orthogonal matrix. If we apply the QR method with double implicit shift with $tol = 10^{-6}$ to A, then after 4 iterations, we obtain a real Schur form $S^TAS = U$. Next, we use the technique discussed prior to Example 2.1.1 to find corresponding eigenvectors. We denote the matrix of eigenvectors by X. Consider the matrix

$$P = \begin{bmatrix} 4 & 5 & 9 & 5 & 0 & 9 & 4 \\ 2 & 5 & 1 & 9 & 7 & 2 & 8 \\ 5 & 3 & 5 & 1 & 9 & 3 & 3 \\ 9 & 10 & 4 & 8 & 6 & 4 & 4 \\ 9 & 5 & 3 & 8 & 7 & 5 & 5 \\ 1 & 3 & 9 & 8 & 7 & 6 & 5 \\ 9 & 1 & 5 & 1 & 0 & 8 & 3 \end{bmatrix}$$

If we use Algorithm 2.3.1 to the perturbed matrix $A_1 = A + P/1000$ with $X_0 = X$, $\Lambda_0 = X^{-1}AX$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 3 iterations we obtain a block diagonalization $Z^{-1}A_1Z = D_1$, whereas if we apply the QR method with double implicit shift to S^TA_1S, then after 6 iterations we obtain a real Schur form $Q_1^T(S^TA_1S)Q_1 = R_1$.

Although this method can be applied to find eigenvalues of some matrices (Example 2.3.1,) in general it does not work for all matrices. For example, if a matrix has equal eigenvalues, or the size of the matrix is large, then the method does not work (Example 2.3.2.) In all of these cases, two 1×1 diagonal blocks approach each other, but as the iterations continue, the rate at which they approach each other gets slower and slower. However if by another method, a block diagonalization $A = X\Lambda X^{-1}$ can be computed, then we can use Λ as the starting guess block diagonal matrix, and X as the starting guess invertible matrix to compute a block diagonalization of the

perturbed matrix $A + \epsilon E$, where E is an arbitrary matrix, and ϵ is a small scalar (Example 2.3.3 and 2.3.4.)

2.4 Convergence of Block Diagonalization of a Matrix

In Section 2.1, we developed Algorithm 2.14 to block diagonalize a nondefective matrix. Here we will examine the speed of convergence of Algorithm 2.14.

Let A be an $n \times n$ nondefective matrix, and let $A = Y\Lambda Y^{-1}$ be a block diagonalization of A, where $\Lambda = \text{diag}(\Lambda_1, \Lambda_2, \ldots, \Lambda_k)$ is a block diagonal matrix, and $Y = [Y_1, Y_2, \ldots, Y_k]$ is an associated block column matrix such that $AY_j = Y_j\Lambda_j$.

Define $X = Y(I + E)$, $X^{new} = X(I + F)$, where E is an $n \times n$ perturbation matrix such that $\text{Diag}(X^{-1}AX) = D = \text{diag}(D_1, D_2, \ldots, D_k)$ is an approximate block diagonalization of A, D_i, and D_j for $i \neq j$ have distinct eigenvalues, and $F = (F_{ij})$ is a $k \times k$ block matrix such that F_{jj} is a square zero matrix, and F_{ij} for $i \neq j$ is the solution P to the matrix equation $PD_j - D_iP = (X^{-1}AX)_{ij}$. Clearly $E = Y^{-1}X - I$. Assume X is close to Y. We will show that $||X^{new} - Y|| = O(||E||^2) = O(||X - Y||^2)$, and $||\text{Diag}(X^{-1}AX) - \Lambda|| \leq O(||X - Y||^2)$. That means the block diagonalization method converges locally quadratically.

Theorem 2.4.1 *Let $A \in \mathbb{R}^{n\times n}$ be a nondefective matrix and suppose $A = Y\Lambda Y^{-1}$ is a block diagonalization of A, where $\Lambda = diag(\Lambda_1, \Lambda_2, \ldots, \Lambda_k)$ is a block diagonal matrix, and $Y = [Y_1, Y_2, \ldots, Y_k]$ is an associated block column matrix such that $AY_j = Y_j\Lambda_j$, and Λ_i, and Λ_j for $i \neq j$ have distinct eigenvalues. Suppose the perturbation matrix $E \in \mathbb{R}^{n\times n}$ is sufficiently small to ensure that X is close to Y in the equation $X = Y(I + E)$, and $Diag(X^{-1}AX) = D$ is an approximate block diagonalization of A. Define $X^{new} = X(I + F)$, where F_{ij} for $i = j$ is a square zero matrix, and for $i \neq j$ is the solution P to the matrix equation $PD_j - D_iP = (X^{-1}AX)_{ij}$.*

Then

$$||Diag\left(X^{-1}AX\right) - \Lambda|| \leq O\left(||E||^2\right), \quad \textit{and} \quad ||X^{new} - Y|| = O\left(||E||^2\right).$$

Proof of Theorem 2.4.1 Assume that if $|p_{ij}| \leq |q_{ij}|$ for $i, j = 1, 2, \ldots, n$, then $||P|| \leq ||Q||$. Now

$$\begin{aligned} X^{-1}AX &= (I+E)^{-1} Y^{-1} AY (I+E) \\ &= (I+E)^{-1} \Lambda (I+E). \end{aligned}$$

Since $(I+E)^{-1} = I - E + E^2 (I+E)^{-1}$, so

$$\begin{aligned} X^{-1}AX &= \left[I - E + E^2 (I+E)^{-1}\right] \Lambda (I+E) \\ &= \Lambda + \Lambda E - E\Lambda - E\Lambda E + E^2 (I+E)^{-1} \Lambda (I+E). \end{aligned}$$

Let $L = -E\Lambda E + E^2 (I+E)^{-1} \Lambda (I+E)$. Then

$$\begin{aligned} ||L|| &\leq || - E\Lambda E|| + ||E^2 (I+E)^{-1} \Lambda (I+E)|| \\ &\leq ||\Lambda||\, ||E||^2 + \frac{||\Lambda||\, ||E||^2 (1 + ||E||)}{1 - ||E||} \\ &= \frac{2\,||\Lambda||\, ||E||^2}{1 - ||E||}. \end{aligned}$$

Since $X^{-1}AX = \Lambda + \Lambda E - E\Lambda + L$, so

$$\text{Diag}\left(X^{-1}AX\right) = \Lambda + \text{Diag}\left(\Lambda E - E\Lambda\right) + \text{Diag}(L). \tag{2.28}$$

Next normalize Y such that the diagonal blocks of $Y^{-1}X$ are identity matrices. That is $\text{Diag}(Y^{-1}X) - I = 0$. Let $Z = Y^{-1}$, then from the relation $E = Y^{-1}X - I$, we have

$$E_{ij} = \begin{cases} 0 & \text{for} \quad i = j, \\ Z_i^T X_j & \text{for} \quad i \neq j, \end{cases}$$

and so, if $C = \Lambda E - E\Lambda$, then

$$C_{ij} = \begin{cases} 0 & \text{for} \quad i = j, \\ \Lambda_i E_{ij} - E_{ij}\Lambda_j & \text{for} \quad i \neq j. \end{cases}$$

Thus $\text{Diag}(\Lambda E - E\Lambda) = \text{Diag}(C) = 0$, and from (2.28) we have $\text{Diag}(X^{-1}AX) - \Lambda = \text{Diag}(L)$. Hence

$$\begin{aligned} ||\text{Diag}\left(X^{-1}AX\right) - \Lambda|| &= ||\text{Diag}(L)|| \\ &\leq ||L|| \leq O\left(||E||^2\right), \end{aligned} \tag{2.29}$$

which proves the first part of Theorem 2.4.1.

Since $\text{Diag}\left(X^{-1}AX\right) = D = \text{diag}(\, D_1, D_2, \ldots, D_k\,)$, and $F = (F_{ij})$ is a $k \times k$ block matrix, where $F_{jj} = 0$, and for $i \neq j$, F_{ij} is the solution B to the matrix equation $BD_j - D_iB = (X^{-1}AX)_{ij}$, so $G = FD - DF$, where

$$\begin{aligned} G_{ij} &= \begin{cases} 0 & \text{for } \; i = j, \\ F_{ij}D_j - D_iF_{ij} & \text{for } \; i \neq j, \end{cases} \\ &= \begin{cases} 0 & \text{for } \; i = j, \\ (X^{-1}AX)_{ij} & \text{for } \; i \neq j. \end{cases} \end{aligned}$$

Hence $FD - DF = G = \text{nondiag}\left(X^{-1}AX\right)$, where $\text{nondiag}(M)$ is a block matrix formed from the block matrix M by replacing the diagonal blocks by zero matrices. Since $X^{-1}AX = \Lambda + \Lambda E - E\Lambda + L$, hence $\text{nondiag}(X^{-1}AX) = \text{nondiag}(\, \Lambda + \Lambda E - E\Lambda + L\,)$. That is

$$FD - DF = \text{nondiag}\,(\Lambda E - E\Lambda + L)\,. \tag{2.30}$$

For $i \neq j$, taking the (i, j) block on both sides of (2.30), we obtain

$$F_{ij}D_j - D_iF_{ij} \;=\; -(E_{ij}\Lambda_j - \Lambda_iE_{ij}) + L_{ij}. \tag{2.31}$$

Let $H_{ij} = E_{ij}\Lambda_j - \Lambda_iE_{ij}$, and let $D_j = U\Sigma U^{-1}$, and $D_i = V\Omega V^{-1}$ be diagonalizations of D_j, and D_i respectively. Substituting these values in (2.31), and then simplifying, we have

$$V^{-1}F_{ij}U\Sigma - \Omega V^{-1}F_{ij}U = -V^{-1}H_{ij}U + V^{-1}L_{ij}U.$$

Let $W = V^{-1}F_{ij}U$. Then the above equation reduces to $W\Sigma - \Omega W = -V^{-1}H_{ij}U + V^{-1}L_{ij}U$. Now solving for $W = (w_{lm})$, we get $w_{lm} = -p_{lm} + q_{lm}$, where $p_{lm} = \frac{(V^{-1}H_{ij}U)_{lm}}{\sigma_m - \omega_l}$, $q_{lm} = \frac{(V^{-1}L_{ij}U)_{lm}}{\sigma_m - \omega_l}$, $l = 1, 2, \ldots, m_i$, and $m = 1, 2, \ldots, m_j$. Let $P = (p_{lm})$, and $Q = (q_{lm})$. Then simplifying the relation $W = -P + Q$, we obtain

$$F_{ij} = -VPU^{-1} + VQU^{-1}. \tag{2.32}$$

Next, let $\Lambda_j = U_1\Sigma_1U_1^{-1}$, and $\Lambda_i = V_1\Omega_1V_1^{-1}$ be diagonalizations of Λ_j, and Λ_i respectively. Using (2.29), we have $||\Lambda_t - D_t|| \leq O(||E||^2)$, where $t = i, j$. So there exist $\Theta_1, \Theta_2, \Gamma_1$, and Γ_2 such that $U_1 = U + \Theta_1$, $\Sigma_1 = \Sigma + \Gamma_1$, $V_1 = V + \Theta_2$, and $\Omega_1 = \Omega + \Gamma_2$, and $||\Theta_t|| \leq O(||E||^2)$ and $||\Gamma_t|| \leq O(||E||^2)$, where $t = 1, 2$. Now

$$\begin{aligned}
\Lambda_j &= (U + \Theta_1)(\Sigma + \Gamma_1)(U + \Theta_1)^{-1} \\
&= (U\Sigma + U\Gamma_1 + \Theta_1\Sigma + \Theta_1\Gamma_1)U^{-1}\left(I + \Theta_1U^{-1}\right)^{-1} \\
&= \left(U\Sigma U^{-1} + U\Gamma_1U^{-1} + \Theta_1\Sigma U^{-1} + \Theta_1\Gamma_1U^{-1}\right)\left(I - \Theta_1U^{-1}\left(I + \Theta_1U^{-1}\right)^{-1}\right) \\
&= U\Sigma U^{-1} + \Delta_1,
\end{aligned}$$

where $\Delta_1 = U\Gamma_1U^{-1} + \Theta_1\Sigma U^{-1} + \Theta_1\Gamma_1U^{-1} - (U\Sigma U^{-1} + U\Gamma_1U^{-1} + \Theta_1\Sigma U^{-1} + \Theta_1\Gamma_1U^{-1})\Theta_1U^{-1}(I + \Theta_1U^{-1})^{-1}$. Hence

$$\begin{aligned}
||\Delta_1|| &\leq ||U\Gamma_1U^{-1}|| + ||\Theta_1\Sigma U^{-1}|| + ||\Theta_1\Gamma_1U^{-1}|| + ||U\Sigma U^{-1} + U\Gamma_1U^{-1} + \Theta_1\Sigma U^{-1} \\
&\quad + \Theta_1\Gamma_1U^{-1}||\,||\Theta_1U^{-1}||\,||\left(I + \Theta_1U^{-1}\right)^{-1}|| \\
&\leq O\left(||E||^2\right).
\end{aligned}$$

Similarly, $\Lambda_i = V\Omega V^{-1} + \Delta_2$ with $||\Delta_2|| \leq O(||E||^2)$. Since $H_{ij} = E_{ij}\Lambda_j - \Lambda_iE_{ij}$, so

$$\begin{aligned}
H_{ij} &= E_{ij}\left(U\Sigma U^{-1} + \Delta_1\right) - \left(V\Omega V^{-1} + \Delta_2\right)E_{ij} \\
V^{-1}H_{ij}U &= V^{-1}E_{ij}U\Sigma - \Omega V^{-1}E_{ij}U + V^{-1}(E_{ij}\Delta_1 - \Delta_2E_{ij})U.
\end{aligned} \tag{2.33}$$

Taking the (l, m) entry on both sides of (2.33), we have

$$\left(V^{-1}H_{ij}U\right)_{lm} = \left(V^{-1}E_{ij}U\right)_{lm}\sigma_m - \omega_l\left(V^{-1}E_{ij}U\right)_{lm} + \left(V^{-1}(E_{ij}\Delta_1 - \Delta_2E_{ij})U\right)_{lm}.$$

After simplifying the last equation, we get $p_{lm} = \dfrac{(V^{-1}H_{ij}U)_{lm}}{\sigma_m - \omega_l} = \left(V^{-1}E_{ij}U\right)_{lm} + s_{lm}$, where $s_{lm} = \dfrac{(V^{-1}(E_{ij}\Delta_1 - \Delta_2 E_{ij})U)_{lm}}{\sigma_m - \omega_l}$. Let $S = (s_{lm})$, then $P = V^{-1}E_{ij}U + S$, and with this value of P (2.32) reduces to

$$\begin{aligned} F_{ij} &= -VV^{-1}E_{ij}UU^{-1} + V(-S+Q)U^{-1} \\ &= -E_{ij} + R_{ij}, \end{aligned}$$

where $R_{ij} = V(-S+Q)U^{-1}$. Since $s_{lm} = \dfrac{(V^{-1}(E_{ij}\Delta_1 - \Delta_2 E_{ij})U)_{lm}}{\sigma_m - \omega_l}$, and $q_{lm} = \dfrac{(V^{-1}L_{ij}U)_{lm}}{\sigma_m - \omega_l}$, so

$$\begin{aligned} ||R_{ij}|| &\leq ||V|| \left(\frac{||V^{-1}||(||E_{ij}||\,||\Delta_1|| + ||\Delta_2||\,||E_{ij}||)||U||}{\min|\,\sigma_m - \omega_l\,|} + \frac{||V^{-1}||\,||L_{ij}||\,||U||}{\min|\,\sigma_m - \omega_l\,|} \right) ||U^{-1}|| \\ &\leq O\left(||E||^2\right). \end{aligned}$$

Thus $F = -E + R$, where $R = (R_{ij})$ with $||R|| = O(||E||^2)$. Next

$$\begin{aligned} X^{new} &= X(I+F) \\ &= Y(I+E)(I-E+R) \\ &= Y\left(I - E^2\right) + Y(I+E)R \\ X^{new} - Y &= -YE^2 + Y(I+E)R. \end{aligned}$$

Hence $||X^{new} - Y|| = O(||E||^2)$. This proves the second part of Theorem 2.4.1. □

The result of the above theorem implies that the block diagonalization of a non-defective matrix converges locally quadratically.

2.5 Block Schur Decomposition of a Matrix

The iterative methods we developed so far to find the eigenvalues and corresponding eigenvectors of a matrix have a little practical interests, unless we have a good

approximation to the matrix of eigenvectors, and in some methods a good approximation to the eigenvalues as well. Here we will find an iterative method to compute a block Schur decomposition of a matrix of A.

Let A be an $n \times n$ complex matrix. Let $A = SUS^H$ be a block Schur decomposition of A, where U is a $k \times k$ block upper triangular matrix whose (i, j) block is an $m_i \times m_j$ matrix, and $S = [\, S_1, S_2, \ldots, S_k \,]$ is a compatible block column unitary matrix such that $AS_j = \sum_{i=1}^{j} S_i U_{ij}$. It can always be arranged so that the eigenvalues of U_{ii}, and U_{jj} for $i \neq j$ are distinct. Before we derive the necessary equations to find a block Schur decomposition of a matrix A, we discuss the effect of perturbations in the elements of A on the columns of the unitary matrix S, and the elements of the upper triangular matrix U in its Schur decomposition $S^H AS = U$. That is given an arbitrary matrix E, and for a sufficiently small ϵ, we will show in the next theorem that there exist continuously differentiable functions $S(\epsilon)$, and $U(\epsilon)$ with $S(0) = S$, and $U(0) = U$ such that $(A + \epsilon E)S(\epsilon) = S(\epsilon)U(\epsilon)$.

Theorem 2.5.1 *Let A be an $n \times n$ complex matrix. Suppose $S^H AS = U$ is a block Schur decomposition of A such that the diagonal blocks U_{ii}, and U_{jj} for $i \neq j$ have distinct eigenvalues. Then given an arbitrary $n \times n$ complex matrix E, and for a sufficiently small ϵ, there exist continuously differentiable functions $S(\epsilon)$, and $U(\epsilon)$ with $S(0) = S$, and $U(0) = U$ such that the following matrix equation holds:*

$$(A + \epsilon E)\, S(\epsilon) = S(\epsilon)U(\epsilon). \tag{2.34}$$

Proof of Theorem 2.5.1 First we discuss the effect of perturbations in the elements of A on the columns of the unitary matrix S in its Schur decomposition $S^H AS = U$. The columns $s_1, s_2, \ldots, s_n$ of S form a basis for $\mathbb{C}^n$. Given an arbitrary $n \times n$ matrix E, and for a sufficiently small ϵ, let $S(\epsilon)^{-1}(A + \epsilon E)S(\epsilon) = U(\epsilon)$ be a triangular decomposition of $A + \epsilon E$ by an invertible matrix $S(\epsilon)$. Let $D = S^H(A + \epsilon E)S$,

and let $Q(\epsilon)^{-1}DQ(\epsilon) = U(\epsilon)$ be the triangular decomposition of D by the invertible matrix $Q(\epsilon)$. Then we have $s_i(\epsilon) = Sq_i(\epsilon)$, $i = 1, \ldots, n$. The form of the matrix D is illustrated for a 7×7 matrix with $k = 5$, such that $\dim(U_{11}) = \dim(U_{22}) = 2$, and $\dim(U_{jj}) = 1$, $j = 3, 4, 5$:

$$D = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} & u_{16} & u_{17} \\ & u_{11} & u_{23} & u_{24} & u_{25} & u_{26} & u_{27} \\ & & u_{33} & u_{34} & u_{35} & u_{36} & u_{37} \\ & & & u_{33} & u_{45} & u_{46} & u_{47} \\ & & & & u_{55} & u_{56} & u_{57} \\ & & & & & u_{66} & u_{67} \\ & & & & & & u_{77} \end{bmatrix} + \epsilon \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} & g_{15} & g_{16} & g_{17} \\ g_{21} & g_{22} & g_{23} & g_{24} & g_{25} & g_{26} & g_{27} \\ g_{31} & g_{32} & g_{33} & g_{34} & g_{35} & g_{36} & g_{37} \\ g_{41} & g_{42} & g_{43} & g_{44} & g_{45} & g_{46} & g_{47} \\ g_{51} & g_{52} & g_{53} & g_{54} & g_{55} & g_{56} & g_{57} \\ g_{61} & g_{62} & g_{63} & g_{64} & g_{65} & g_{66} & g_{67} \\ g_{71} & g_{72} & g_{73} & g_{74} & g_{75} & g_{76} & g_{77} \end{bmatrix},$$

where $g_{ij} = s_i^H E s_j$. Equating the jth column on both sides of $DQ(\epsilon) = Q(\epsilon)U(\epsilon)$, we obtain

$$\sum_{l=i}^{n} u_{il} q_{lj}(\epsilon) + \epsilon \sum_{l=1}^{n} g_{il} q_{lj}(\epsilon) = \sum_{l=1}^{j} q_{il}(\epsilon) u_{lj}(\epsilon), \tag{2.35}$$

$i, j = 1, \ldots, n$. Since $u_j(0) = u_j$, so we must have $q_j(0) = e_j$, which is the jth unit vector. Next we normalize $q_j(\epsilon)$, $j = 1, \ldots, n$ by setting $q_{ij}(\epsilon) = 0$, $i = 1, \ldots, j-1$, and taking 1 as the largest element. For sufficiently small ϵ, claim that $q_{11}(\epsilon) = 1$, which is the 1st component of $q_1(\epsilon)$. If not, let $q_{n1}(\epsilon) = 1$. Then from (2.35) we get

$$u_{11}(\epsilon) - u_{nn} = \epsilon \sum_{l=1}^{n} g_{nl} q_{l1}(\epsilon).$$

Now letting $\epsilon \to 0$, we find $u_{11} - u_{nn} = 0$, a contradiction. Next we will show that $|q_{n1}(\epsilon)| \leq M_{n1}\epsilon$ as $\epsilon \to 0$, for some $M_{n1} > 0$. Since $|q_{i1}(\epsilon)| \leq 1$, so from (2.35), we obtain

$$|u_{11}(\epsilon) - u_{nn}|\,|q_{n1}(\epsilon)| \leq \epsilon \sum_{l=1}^{n} |g_{nl}|.$$

Let $|u_{11}(\epsilon) - u_{nn}| \geq \frac{1}{2}|u_{11} - u_{nn}|$, then $|q_{n1}(\epsilon)| \leq M_{n1}\epsilon$, where

$$M_{n1} = \frac{2 \sum_{l=1}^{n} |g_{nl}|}{|u_{11} - u_{nn}|}.$$

Using the above arguments to the components q_{i1}, $i = 3, \ldots, n-1$ in the decreasing order of the row index i, we can show that $q_{i1}(\epsilon) \neq 1$, and $|q_{i1}(\epsilon)| \leq M_{i1}\epsilon$, where $M_{i1} > 0$. Again from (2.35), we have

$$(u_{11}(\epsilon) - u_{11})q_{11}(\epsilon) = \sum_{l=2}^{n} u_{1l}q_{l1}(\epsilon) + \epsilon \sum_{l=1}^{n} g_{1l}q_{l1}(\epsilon),$$

$$(u_{11}(\epsilon) - u_{11})q_{21}(\epsilon) = \sum_{l=3}^{n} u_{2l}q_{l1}(\epsilon) + \epsilon \sum_{l=1}^{n} g_{2l}q_{l1}(\epsilon).$$

To make the above two equations consistent we take $q_{21}(\epsilon) = 0$. Using similar arguments, and considering the elements in the decreasing order of the row index i, we can show that $|q_{ij}(\epsilon)| \leq M_{ij}\epsilon$ for some $M_{ij} > 0$, $j = 2, \ldots, n-1$, $i = j+1, \ldots, n$. To show $|q_{ij}(\epsilon)| \leq M_{ij}\epsilon$, we use the results $|q_{il}(\epsilon)| \leq M_{il}\epsilon$, $l = 1, \ldots, j-1$, and $|q_{lj}(\epsilon)| \leq M_{lj}\epsilon$, $l = i+1, \ldots, n$. To keep the consistency, we take $q_{43}(\epsilon) = 0$. Let

$$M = \max_{1 \leq j < n,\ j < i \leq n} M_{ij}.$$

Then $|q_{ij}(\epsilon)| \leq M\epsilon$, $j = 1, \ldots, n-1$, $i = j+1, \ldots, n$.

Thus $q_{ij}(\epsilon)$ are of order ϵ for $j = 1, \ldots, n-1$, $i = j+1, \ldots, n$, except $q_{21}(\epsilon)$, and $q_{43}(\epsilon)$, which are equal to zero. Next from (2.35), we have

$$(u_{jj}(\epsilon) - u_{ii})q_{ij}(\epsilon) + \sum_{l=1}^{j-1} q_{il}(\epsilon)u_{lj}(\epsilon) - \sum_{l=i+1}^{n} u_{il}q_{lj}(\epsilon) - \epsilon g_{ij} = \epsilon \sum_{l=j+1}^{n} g_{il}q_{lj}(\epsilon).$$

The term on the right is of order ϵ^2, so

$$\left| q_{ij}(\epsilon) - \frac{\epsilon}{u_{jj} - u_{ii}} \left(g_{ij} + \sum_{l=i+1}^{n} u_{il}q'_{lj}(0) - \sum_{l=1}^{j-1} q'_{il}(0)u_{lj} \right) \right| = O(\epsilon^2).$$

Next we discuss the effect of perturbations in the elements of A on the elements of the upper triangular matrix U in its Schur decomposition $S^H AS = U$. As in the earlier discussion, let $D = S^H(A + \epsilon E)S$, and $Q(\epsilon)^{-1}DQ(\epsilon) = U(\epsilon)$ be the triangular decomposition of D by the invertible matrix $Q(\epsilon)$. To illustrate, we consider the

same 7×7 matrix D with $k = 5$. Here we will use the results of perturbations of the columns of $Q(\epsilon)$. Putting $j = 1,\ i = 1$, and $j = 2,\ i = 1, 2$ in (2.35), we obtain

$$\sum_{l=1}^{n} u_{1l} q_{l1}(\epsilon) + \epsilon \sum_{l=1}^{n} g_{1l} q_{l1}(\epsilon) = u_{11}(\epsilon),$$

$$\sum_{l=2}^{n} u_{1l} q_{l2}(\epsilon) + \epsilon \sum_{l=2}^{n} g_{1l} q_{l2}(\epsilon) = u_{12}(\epsilon),$$

$$\sum_{l=2}^{n} u_{2l} q_{l2}(\epsilon) + \epsilon \sum_{l=2}^{n} g_{2l} q_{l2}(\epsilon) = u_{11}(\epsilon).$$

After simplifying the above equations, we get

$$u_{11}(\epsilon) - \left(u_{11} + \epsilon g_{11} + \sum_{l=3}^{n} u_{1l} q_{l1}(\epsilon) \right) = \epsilon \sum_{l=3}^{n} g_{1l} q_{l1}(\epsilon),$$

$$u_{12}(\epsilon) - \left(u_{12} + \epsilon g_{12} + \sum_{l=3}^{n} u_{1l} q_{l2}(\epsilon) \right) = \epsilon \sum_{l=3}^{n} g_{1l} q_{l2}(\epsilon),$$

$$u_{11}(\epsilon) - \left(u_{11} + \epsilon g_{22} + \sum_{l=3}^{n} u_{2l} q_{l2}(\epsilon) \right) = \epsilon \sum_{l=3}^{n} g_{2l} q_{l2}(\epsilon).$$

For sufficiently small ϵ, let $|u_{11}(\epsilon) - u_{ii}| \geq \frac{1}{2}|u_{11} - u_{ii}|,\ i = 3, 5, \ldots, n$. Then the terms on the right of the above equations will be of order ϵ^2, and we have

$$\left| u_{11}(\epsilon) - \left(u_{11} + \epsilon g_{11} + \epsilon \sum_{l=3}^{n} u_{1l} q'_{l1}(0) \right) \right| = O(\epsilon^2),$$

$$\left| u_{12}(\epsilon) - \left(u_{12} + \epsilon g_{12} + \epsilon \sum_{l=3}^{n} u_{1l} q'_{l2}(0) \right) \right| = O(\epsilon^2),$$

$$\left| u_{11}(\epsilon) - \left(u_{11} + \epsilon g_{22} + \epsilon \sum_{l=3}^{n} u_{2l} q'_{l2}(0) \right) \right| = O(\epsilon^2).$$

Thus perturbations in u_{11}, and u_{12} are of order ϵ as $\epsilon \to 0$. Due to Theorem 2.1.1, it is always possible to find a sufficiently small ϵ, such that $|u_{jj}(\epsilon) - u_{ii}| \geq \frac{1}{2}|u_{jj} - u_{ii}|$ for $i \neq j,\ i, j = 1, 3, 5, \ldots, n$. So after simplifying (2.35), we can show that for $i \leq j,\ j = 3, 5, \ldots, n$,

$$u_{ij}(\epsilon) - \left(u_{ij} + \epsilon g_{ij} + \sum_{l=j+1}^{n} u_{il} q_{lj}(\epsilon) - \sum_{l=1}^{i-1} q_{il}(\epsilon) u_{lj} \right) = O(\epsilon^2).$$

Hence

$$\left| u_{ij}(\epsilon) - \left(u_{ij} + \epsilon g_{ij} + \epsilon \sum_{l=j+1}^{n} u_{il} q'_{lj}(0) - \epsilon \sum_{l=1}^{i-1} q'_{il}(0) u_{lj} \right) \right| = O(\epsilon^2).$$

Thus the perturbation in u_{ij} is of order ϵ as $\epsilon \to 0$. Hence $U(\epsilon)$, and $S(\epsilon)$ are differentiable functions of ϵ, and their first derivatives are continuous. This completes the proof of the theorem. □

Now we are ready to derive the necessary equations to find a block Schur decomposition $A = SUS^H$. Given an $n \times n$ complex matrix E, and for a sufficiently small ϵ, let $S(\epsilon)$, and $U(\epsilon)$ be continuously differentiable functions of ϵ such that $S(0) = S$, and $U(0) = U$. Differentiating (2.34) with respect to ϵ and then putting $\epsilon = 0$, we obtain

$$AS'(0) + ES = S'(0)U + SU'(0)$$

$$S^H AS'(0) + S^H ES = S^H S'(0)U + U'(0). \tag{2.36}$$

Since $S^H A = US^H$, so (2.36) gives

$$US^H S'(0) + S^H ES = S^H S'(0)U + U'(0). \tag{2.37}$$

Taking the jth column block on both sides of (2.37), we have

$$US^H S'_j(0) + S^H ES_j = S^H S'(0)U_j + U'_j(0). \tag{2.38}$$

Next consider the following expressions for $S_j(\epsilon)$:

$$S_j(\epsilon) = \sum_{i=1}^{k} S_i B_{ij}(\epsilon), \tag{2.39}$$

where $B_{ij}(\epsilon)$ is an $m_i \times m_j$ matrix. We normalize (2.39) by taking $B_{jj}(\epsilon) = I$, and $B_{ij}(\epsilon) = 0$ for $i < j$. Differentiating (2.39) with respect to ϵ and then setting $\epsilon = 0$, we get

$$S'_j(0) = \sum_{i=j+1}^{k} S_i B'_{ij}(0). \tag{2.40}$$

Since S is unitary, so

$$S_i^H S_j = \begin{cases} 0 & \text{for } i \neq j, \\ I & \text{for } i = j. \end{cases} \tag{2.41}$$

Using (2.40), and (2.41) we get $S^H S'(0) = B'(0)$. So (2.38) reduces to

$$UB'_j(0) + S^H E S_j = B'(0)U_j + U'_j(0),$$

where $B'_j(0)$ is the jth column block of $B'(0)$. Hence for each j, we have the following two matrix equations:

$$\sum_{l=j+1}^{k} U_{il} B'_{lj}(0) + S_i^H E S_j = \sum_{l=1}^{i-1} B'_{il}(0) U_{lj} + U'_{ij}(0) \quad \text{for } i \leq j,\ j = 1, 2, \ldots, k.$$

$$\sum_{l=i}^{k} U_{il} B'_{lj}(0) + S_i^H E S_j = \sum_{l=1}^{j} B'_{il}(0) U_{lj} \quad \text{for } i > j,\ j = 1, \ldots, k-1.$$

The last two equations can be rewritten as:

$$U'_{ij}(0) = S_i^H E S_j + \sum_{l=j+1}^{k} U_{il} B'_{lj}(0) - \sum_{l=1}^{i-1} B'_{il}(0) U_{lj} \quad \text{for } i \leq j,\ j = 1, 2, \ldots, k. \tag{2.42}$$

$$B'_{ij}(0)U_{jj} - U_{ii}B'_{ij}(0) = S_i^H E S_j + \sum_{l=i+1}^{k} U_{il} B'_{lj}(0) - \sum_{l=1}^{j-1} B'_{il}(0) U_{lj} \text{ for } i > j,\ j = 1, \ldots, k-1. \tag{2.43}$$

Solution of (2.42) is dependent on the solution of (2.43). So we focus on how to find the solution of (2.43). From the right hand side of (2.43), it is clear that to find $B'_{ij}(0)$, we need entries below $B'_{ij}(0)$ on the jth block column; namely $B'_{i+1j}(0), \ldots, B'_{kj}(0)$, and entries on the left of $B'_{ij}(0)$ on the ith block row; namely $B'_{i1}(0), \ldots, B'_{ij-1}(0)$. Pictorially these can be described with zero suppressed in $B'_{ij}(0)$ as:

$$\begin{bmatrix} & & & & \\ & & & & \\ & & & & \\ B'_{i1} & \cdots & B'_{ij-1} & B'_{ij} & \\ \longleftarrow & \longleftarrow & \longleftarrow & & \\ & & & \downarrow\ B'_{i+1j} & \\ & & & \downarrow\ \ \vdots & \\ & & & \downarrow\ B'_{kj} & \end{bmatrix}$$

That is to determine B'_{ij}, we need entries just above the arrowheads directing towards the left, and just on the right of the downwards arrowheads.

Here we will give a sketch to find B'_{ij}. Let $C = S_i^H E S_j$. If $i < k$, then $C = C + \sum_{l=i+1}^{k} U_{il} B'_{lj}$, and if $j > 1$, then $C = C - \sum_{l=1}^{j-1} B'_{il} U_{lj}$. Next we need to solve the matrix equation $B'_{ij} U_{jj} - U_{ii} B'_{ij} = C$ for B'_{ij}. To solve this equation, let $P = B'_{ij}$, $F = U_{jj}$, and $G = U_{ii}$. Then it becomes $GP - PF = R$, where $R = -C$. Since F, and G are upper triangular matrices, so using the following method (detail is in [Gol90, page 387]), we can solve the matrix equation $GP - PF = R$ for P. Let $R = [r_1, r_2, \ldots, r_{m_j}]$, and $P = [p_1, p_2, \ldots, p_{m_j}]$ be column partitionings, then solve

$$(G - f_{ll} I) p_l = r_l + \sum_{m=1}^{l-1} f_{ml} p_m$$

for p_l, $l = 1, \ldots, m_j$. Once we obtain P, then $B'_{ij} = P$ for $i > j$, $j = 1, 2, \ldots, k-1$.

2.6 An Algorithm for Block Schur Decomposition of a Matrix

In Section 2.5, we derived the necessary equations to find a block Schur decomposition of a matrix. Here we plan to develop an algorithm using Armijo's rule from optimization theory to find a block Schur decomposition of a matrix.

Let A be an $n \times n$ complex matrix. Suppose $A = SUS^H$ is a block Schur decomposition of A, where U is a $k \times k$ block upper triangular matrix such that U_{ii}, and U_{jj} for $i \neq j$ have distinct eigenvalues, and $S = [S_1, S_2, \ldots, S_k]$ is a compatible block column unitary matrix such that $AS_j = \sum_{i=1}^{j} S_i U_{ij}$. Given an arbitrary complex matrix E, and for a sufficiently small ϵ, let $S(\epsilon)$, and $U(\epsilon)$ be analytic functions of ϵ, such that $S(0) = S$, $U(0) = U$, and

$$(A + \epsilon E) S(\epsilon) = S(\epsilon) U(\epsilon). \tag{2.44}$$

A first order Taylor Expansion gives

$$\begin{aligned} S(\epsilon) &\approx S(0) + \epsilon S'(0), \\ U(\epsilon) &\approx U(0) + \epsilon U'(0). \end{aligned} \tag{2.45}$$

To develop an algorithm, suppose SUS^H is an approximate block Schur decomposition of B. We identify A, and E in (2.44) with SUS^H, and $B - SUS^H$ respectively. Let $S'(0) = SG(S,U)$, where $G_{ij}(S,U)$ is an $m_i \times m_j$ zero matrix for $i \leq j$, and for $i > j$, $G_{ij}(S,U)$ is the solution P to the matrix equation $PU_{jj} - U_{ii}P = C$, where

$$C = S_i^H \left(B - SUS^H\right) S_j + \sum_{l=i+1}^{k} U_{il} G_{lj}(S,U) - \sum_{l=1}^{j-1} G_{il}(S,U) U_{lj}.$$

From (2.37), we have

$$US^H S'(0) + S^H ES = S^H S'(0) U + U'(0).$$

Next substituting the values of E, and $S'(0)$ in the above equation and then simplifying, we obtain

$$U'(0) = UG(S,U) + S^H BS - U - G(S,U)U.$$

Now substituting the values of $U'(0)$, and $S'(0)$ into (2.45) and suppressing zeros in $S(0)$, and $U(0)$, we get

$$\begin{aligned} S(\epsilon) &\approx S\left(I + \epsilon G(S,U)\right), \\ U(\epsilon) &\approx U + \epsilon \left(UG(S,U) + S^H BS - U - G(S,U)U\right). \end{aligned} \tag{2.46}$$

Taking $\epsilon = 1$ in (2.46), we obtain

$$\begin{aligned} S^{new} &= S^{old}\left(I + G\left(S^{old}, U^{old}\right)\right), \\ U^{new} &= U^{old} G\left(S^{old}, U^{old}\right) - \left(S^{old}\right)^H BS^{old} - G\left(S^{old}, U^{old}\right) U^{old}. \end{aligned}$$

If the starting guess unitary matrix S_0 is not a good approximation of Q, where $B = QRQ^H$ is a Schur decomposition of B, then in the update $S^{new} = S^{old}(I + G\left(S^{old}, U^{old}\right))$, the increment $S^{old} G\left(S^{old}, U^{old}\right)$ may be too large. However with a large increment, we may likely have instabilitiy in the updating process, and the algorithm may diverge. To restore the convergence of the algorithm, and to have a

steady change in the values of S^{new}, and U^{new}, we need a small increment in each iteration. To achieve this, we redefine each iterate, and introduce a small positive parameter. Let t be a positive parameter; define $S(t) = S^{old}\left(I + tG\left(S^{old}, U^{old}\right)\right)$, and $U(t) = U^{old} + t\left(U^{old}G\left(S^{old}, U^{old}\right) + \left(S^{old}\right)^{H} BS^{old} - U^{old} - G\left(S^{old}, U^{old}\right)U^{old}\right)$. Then $S(0) = S^{old}$, $S(1) = S^{new}$, $U(0) = U^{old}$, and $U(1) = U^{new}$. Define $f(t) = ||Z(t)||_F$, where $Z(t) = B - S(t)U(t)S(t)^{-1}$. Then $f(0) = ||B - S(0)U(0)S(0)^{H}||_F$, and $f(1) = ||B - S(1)U(1)S(1)^{-1}||_F$. When the starting guess S_0, and U_0 are good approximations of Q, and R in the factorization $B = QRQ^{H}$, then we must have $f(1) \leq f(0)$. As earlier, here our goal is to find a t, $0 < t \leq 1$, for which $f(t) \leq f(0)$ holds. To this end, we use Armijo's rule from optimization theory. In Armijo's rule, we determine t in the following way. Evaluate $f(t)$ at $t = 1, \frac{1}{2}, \frac{1}{4}, \ldots$, stopping when

$$f(t) \leq \left(1 - \frac{t}{2}\right) f(0).$$

As in Section 2.3, to use the above rule, we must have $f'(0) = -f(0)$. So our next aim is to determine $f'(0)$, when $f(t) = ||Z(t)||_F$, and $Z(t) = B - S(t)U(t)S(t)^{-1}$. Suppressing the superscripts of S^{old}, and U^{old} in the definitions of $S(t)$, and $U(t)$ we get

$$\begin{aligned} S(t) &= S\left(I + tG\left(S, U\right)\right), \\ U(t) &= U + t\left(UG\left(S, U\right) + S^{H}BS - U - G\left(S, U\right)U\right). \end{aligned} \tag{2.47}$$

With the above definition of $f(t)$, we will show in the next theorem that $f'(0) = -f(0)$.

Theorem 2.6.1 *Let $A \in \mathbb{C}^{n\times n}$ and suppose $A = SUS^{H}$ is a block Schur decomposition of A, where U is a $k \times k$ block upper triangular matrix such that U_{ii}, and U_{jj} for $i \neq j$ have distinct eigenvalues, and $S = [\,S_1, S_2, \ldots, S_k\,]$ is a compatible block column unitary matrix. If we define $U(t)$, $S(t)$ as in (2.47), $Z(t) = A - S(t)U(t)S(t)^{-1}$, and $f(t) = ||Z(t)||_F$, then $f'(0) = -f(0)$.*

Proof of Theorem 2.6.1 The result (2.18) of Lemma 2.3.1 gives

$$f(0)\frac{d}{dt}f(t)|_{t=0} = \text{trace}\left(Z(0)^H \frac{d}{dt}Z(t)|_{t=0}\right). \tag{2.48}$$

Differentiating the expression for $Z(t)$ with respect to t, and then evaluating the derivative at $t = 0$, we have

$$\begin{aligned}\frac{d}{dt}Z(t)|_{t=0} &= -\frac{d}{dt}S(t)|_{t=0}U(0)S(0)^{-1} - S(0)\frac{d}{dt}U(t)|_{t=0}S(0)^{-1} \\ &\quad + S(0)U(0)S(0)^{-1}\frac{d}{dt}S(t)|_{t=0}S(0)^{-1}. \end{aligned}\tag{2.49}$$

Next differentiating $U(t)$, and $S(t)$ with respect to t, and then evaluating the derivatives at $t = 0$, we obtain

$$\frac{d}{dt}U(t)|_{t=0} = UG(S,U) + S^H AS - U - G(S,U)U, \quad \text{and} \quad \frac{d}{dt}S(t)|_{t=0} = SG(S,U).$$

Using these values and values of $U(0)$, and $S(0)$ in (2.49), we get

$$\begin{aligned}\frac{d}{dt}Z(t)|_{t=0} &= -SG(S,U)US^H - S\left(UG(S,U) + S^H AS - U - G(S,U)U\right)S^H \\ &\quad + SUS^H SG(S,U)S^H \\ &= -\left(A - SUS^H\right) \\ &= -Z(0).\end{aligned}$$

Hence $\frac{d}{dt}Z(t)|_{t=0} = -Z(0)$, and with this value (2.48) reduces to

$$\begin{aligned}f(0)\frac{d}{dt}f(t)|_{t=0} &= \text{trace}\left(Z(0)^H(-Z(0))\right) \\ &= -f(0)^2.\end{aligned}$$

So $\frac{d}{dt}f(t)|_{t=0} = -f(0)$, provided $f(0) \neq 0$. □

Thus Armijo's rule can be applied to find a block Schur decomposition of a matrix A and we will compute a block Schur decomposition of A as follows:

Algorithm 2.6.1 (Block Schur Decomposition) Given $A \in \mathbb{C}^{n\times n}$, a tolerance *tol* greater than the unit roundoff, a coalescing tolerance *tol1* greater than the square root of the unit roundoff, an upper triangular matrix U_0, and a unitary matrix S_0, the following algorithm uses (2.47) to compute a block Schur decomposition $A = SUS^H$.

Define $tol2 = 100 tol1$. We break the sketch of the algorithm into several steps.

Step 0: Take $S = S_0$, $U = U_0$. Let NB denote the number of diagonal blocks of U, and let NS denote an array of NB elements whose ith element is the dimension of the ith diagonal block of U.

Step 1: Let $m = NB$. For $i < j$, define $\alpha_{ij} = \min\{\, |\sigma - \omega| : \sigma \in \lambda(U_{ii}),$ and $\omega \in \lambda(U_{jj}) \,\}$. If $\alpha_{ij} \le tol1$, then merge blocks U_{ii} and U_{jj} to form a single block using a unitary transformation to U. Postmultiply S by the same unitary transformation. Update NB^{old}, and NS^{old} to get NB^{new}, and NS^{new}. Try the above coupling procedure for all possible combinations of $1 \le i < j \le m$. Using a unitary transformation arrange diagonal blocks of U in the decreasing order of sizes. Postmultiply S by the same unitary transformation.

Step 2: Construct $G(S,U)$ as follows:

For $i \le j$, take $G_{ij}(S,U) = 0$, which is an $m_i \times m_j$ zero matrix, and for $i > j$, $G_{ij}(S,U)$ can be determined by using the following loop.

$B = AS - SU$

for $j = 1 : NB - 1$

  for $i = NB : -1 : j + 1$

    $C = S_i^H B_j$

    if $i < NB$

      $C = C + \sum_{k=i+1}^{NB} U_{ik} G_{kj}(S,U)$

    end

if $j > 1$

$$C = C - \sum_{k=1}^{j-1} G_{ik}(S,U)U_{kj}$$

end

Solve $PU_{jj} - U_{ii}P = C$ for P

$G_{ij}(S,U) = P$

end

end

Step 3: Let $f(t) = ||Z(t)||_F$, where $Z(t) = A - S(t)U(t)S(t)^{-1}$, $S(t) = S(I + tG(S,U))$, and $U(t) = U + t\left(UG(S,U) + S^H AS - U - G(S,U)U\right)$. Evaluate $f(t)$ at $t = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$, stopping when

$$f(t) \leq \left(1 - \frac{t}{2}\right) f(0). \tag{2.50}$$

Let p be the first value of t for which (2.50) is true.

Step 4: Suppose $\mu = [\, U_{m+1m+1}, \ldots, U_{NBNB}\,]^T$, where $0 \leq m < NB$ with $\dim(U_{ii}) > 1$ for $i = 1, 2, \ldots, m$, and $\dim(U_{ii}) = 1$ for $i = m+1, \ldots, NB$. Define $d = \text{diag}\left(UG(S,U) + S^H AS - U - G(S,U)U\right)(t : n)$, where $t = 1 + \sum_{i=1}^{m} NS(i)$, and $\text{diag}(M)\,(t:n)$ is a vector formed from t through n elements of the vector $\text{diag}(M)$. Our aim is to find two indices il, and iu as follows. Let k be an array with the property that $|\mu_{k(i)}| \leq |\mu_{k(i+1)}|$. Next form the ratio:

$$r_i = \frac{\mu_{k(i+1)} - \mu_{k(i)}}{d_{k(i)} - d_{k(i+1)}},$$

and let $|r_{i_0}| = \min\{\, |r_i| : i = 1, 2, \ldots, n-t,\ \text{Re}(r_i) > 0\ \&\ |r_i| < 1\,\}$. Let $il = t - 1 + \min\{\, k(i_0), k(i_0+1)\,\}$, and $iu = t - 1 + \max\{\, k(i_0), k(i_0+1)\,\}$.

Step 5: Update U, and S as follows:

$$\begin{aligned} U^{new} \;=\; & U^{old} + p\left(U^{old} G\left(S^{old}, U^{old}\right) + \left(S^{old}\right)^H AS^{old}\right. \\ & \left. - U^{old} - G\left(S^{old}, U^{old}\right) U^{old}\right), \end{aligned}$$

$$S^{new} = S^{old}\left(I + pG\left(S^{old}, U^{old}\right)\right).$$

Let $Z = S^{new}U^{new}(S^{new})^{-1}$.

Step 6: Consider the block U_{jj} with $1 \leq j \leq m$, where m is defined as in Step 4. Let $U_{jj} = QRQ^H$ be a Schur decomposition of U_{jj}. Find a unitary matrix P such that $P^H RP = V = D + N$, where $D = \text{diag}(\, V_{11}, V_{22}, \ldots, V_{kk}\,)$, N is the strictly upper triangular part of V, $k \leq m_j$, and $\min\{\, |\sigma - \omega| : \sigma \in \lambda(V_{ii}),$ and $\omega \in \lambda(V_{ll})\,\}$ $> tol2$. Replace U_{jj} by V, and S_j by S_jQP. Update NB^{old}, and NS^{old} to get NB^{new}, and NS^{new}. Try the above decoupling procedure for all j such that $1 \leq j \leq m$.

Step 7: Create a 2×2 diagonal block by merging $U_{il,il}$ and $U_{iu,iu}$ by a unitary transformation. Postmultiply S by the same unitary transformation.

Step 8: Let $S^{old} = QR$ be a QR-factorization of S^{old}. Then take $S^{new} = Q$, and $U^{new} = Q^H ZQ$, where Z is the matrix from Step 5.

Step 9: Compute $f(0) = ||A - Z||_F$ and goto Step 1 until $f(0) < tol$.

Example 2.6.1 If we apply the real version of Algorithm 2.6.1 to the matrix A in Example 2.3.1 with $U_0 = \text{diag}(\, 0, -2, 0\,)$, $S_0 = I$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 9 iterations the upper triangular matrix U_0 converges to

$$U = \begin{bmatrix} -0.88991 & 2.72597 & 1.63393 \\ -0.73813 & -1.11009 & -1.82490 \\ 0 & 0 & 0 \end{bmatrix}.$$

The eigenvalues of U are $\lambda = [\, -1 + 1.41421i, -1 - 1.41421i, 0\,]$. If we use the QR method with double implicit shift to A with the same tolerance, then after 5 iterations we get a real Schur decomposition $A = QRQ^T$, where

$$R = \begin{bmatrix} 0 & 1.7321 & 0 \\ -1.7321 & -2 & 2.4495 \\ 0 & 0 & 0 \end{bmatrix}, \text{ and } Q = \begin{bmatrix} 0.8165 & 0 & -0.5774 \\ -0.4082 & 0.7071 & -0.5774 \\ 0.4082 & 0.7071 & 0.5774 \end{bmatrix}.$$

Next consider the following matrix A_1, which we obtain by perturbing the elements of A:

$$A_1 = \begin{bmatrix} 0.001 & 1.002 & 1.001 \\ -2.002 & -2.003 & -1.001 \\ -2.002 & -1.002 & 0.001 \end{bmatrix}.$$

If we apply the real version of Algorithm 2.6.1 to A_1 with $S_0 = Q$, $U_0 = R$, the tolerance, and the coalescing tolerance are the same, then after 1 iteration we get a block Schur decomposition $P^T A_1 P = \Omega$, whereas if we use the QR method with double implicit shift to $Q^T A_1 Q$, then after 5 iterations we obtain a real Schur form $V^T(Q^T A_1 Q)V = \Gamma$.

Example 2.6.2 If the real version of Algorithm 2.6.1 is applied to the matrix A in Example 2.3.2 with $U_0 = \text{diag}(-2, 1, -1)$, $S_0 = I$, the tolerance, and the coalescing tolerance are as in Example 2.6.1, then after 10 iterations the upper triangular matrix U_0 converges to

$$U = \begin{bmatrix} -1.00104 & -3.13124 & -1.64159 \\ 0 & 0 & -1.22496 \\ 0 & 0 & -0.99896 \end{bmatrix}.$$

If we use the QR method with double implicit shift to A with the same tolerance, then after 6 iterations we obtain a real Schur form:

$$Q^T A Q = \begin{bmatrix} -1.04199 & -3.46359 & -0.69221 \\ 0.00051 & -0.95802 & 1.23323 \\ 0 & 0 & 0 \end{bmatrix}.$$

Example 2.6.3 Consider the following matrices A, and P:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -80 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 236 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -164 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -141 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 243 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -102 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 7 \end{bmatrix}, \text{ and } P = \begin{bmatrix} 2 & 2 & 1 & 1 & 1 & 3 & 3 & 1 \\ 2 & 2 & 1 & 1 & 1 & 1 & 3 & 3 \\ 1 & 1 & 1 & 1 & 3 & 1 & 2 & 2 \\ 1 & 1 & 1 & 2 & 3 & 2 & 3 & 1 \\ 1 & 2 & 1 & 3 & 1 & 2 & 1 & 3 \\ 2 & 2 & 3 & 1 & 1 & 1 & 1 & 3 \\ 3 & 3 & 1 & 2 & 1 & 1 & 2 & 2 \\ 1 & 2 & 3 & 1 & 1 & 2 & 1 & 3 \end{bmatrix}.$$

If we apply the QR method with double implicit shift to A with $tol = 10^{-6}$, then after 15 iterations we get a real Schur form $S^TAS = U$. The eigenvalues of U are $\lambda = [-4, -1, 2, 2, 5, 1, 1, 1]$. Now if the real version of Algorithm 2.6.1 is applied to the perturbed matrix $A_1 = A + P/5000$ with $U_0 = U$, $S_0 = S$, $tol1 = 10^{-4}$, and the tolerance is as above, then after 3 iterations we obtain a block Schur decomposition $Q^TA_1Q = \Omega$, whereas if we apply the QR method with double implicit shift to S^TA_1S, then after 10 iterations we obtain a real Schur form $V^T(S^TA_1S)V = \Gamma$.

Although the real version of Algorithm 2.6.1 works for all matrices A with a diagonal matrix formed from the diagonal elements of A, as the starting upper triangular matrix U_0, and $S_0 = I$, it has some disadvantages. For example, if the size of the matrix is large, then we need to take a very large coalescing tolerance compared to $tol1 = 10^{-4}$ through several iterations at the beginning. However with a large coalescing tolerance, the sizes of some diagonal blocks of U become large compared to the size of A, which undermines the main goal of this algorithm; namely keep the size of the diagonal blocks of U as small as possible. In Example 2.6.1, and 2.6.3 we noticed that if by another method, a real Schur form $A = SUS^T$ can be computed, then we can use U as the starting upper triangular matrix, and S as the starting orthogonal matrix to obtain a block Schur decomposition of a perturbed matrix $A + \epsilon E$ with very rapid convergence, where E is an arbitrary matrix, and ϵ is a small scalar. In the following table for different values of n we give the number of iterations required to obtain block Schur decompositions of $A + \epsilon E$ by the real version of Algorithm 2.6.1 with $U_0 = U$, and $S_0 = S$, and real Schur forms of $S^T(A + \epsilon E)S$ by the QR method with double implicit shift, where $S^TAS = U$ is a real Schur form of the matrix A. Here the original matrices $A = (a_{ij})$ are random matrices, where a_{ij} are integers and $0 \leq a_{ij} \leq 10$, the matrices $E = (e_{ij})$ are random matrices, where e_{ij} are reals and $0 < e_{ij} < 1$, the tolerance $tol = 10^{-6}$, and the coalescing tolerance $tol1 = 10^{-4}$:

size	number of iterations for our algorithm	number of iterations for the QR algorithm with double shift		ϵ
	$A+\epsilon E$	$A+\epsilon E$	$S^T(A+\epsilon E)S$	
10	2	15	19	10^{-2}
20	2	27	28	10^{-2}
30	2	42	45	10^{-2}
40	2	53	56	10^{-2}
50	2	79	70	10^{-2}

Example 2.6.4 If Algorithm 2.6.1 is applied to the matrix

$$A = \begin{bmatrix} 4 & -5 & 0 & 3 & -2 & 1 \\ 0 & -4 & -3 & -5 & -1 & 2 \\ 5 & 4 & -3 & 0 & 5 & -3 \\ 3 & 0 & 5 & 4 & 2 & -1 \\ -6 & 1 & 3 & -4 & 1 & -2 \\ 2 & -3 & -1 & 1 & 5 & -3 \end{bmatrix}$$

with $U_0 = \text{diag}(\ 19,\ 9.5 + 16.45448i,\ -9.5 + 16.45448i,\ -19,\ -9.5 - 16.45448i,\ 9.5 - 16.45448i)$, where the elements of U_0 are the uniformly distributed points on a circle, which includes all Gerschgorin disks for A, $S_0 = I$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 13 iterations we obtain a block Schur decomposition $S^H AS = U$, where $U = D + N$, $D = \text{diag}(\ -2.25911 - 3.43367i,\ 1.83571 - 3.95888i,\ -2.25911 + 3.43367i,\ -7.52532,\ 1.83571 + 3.95888i,\ 7.37213\ )$, and N is the strictly upper triangular part of U. Figure 2.1 shows how the initial eigenvalues of U_0 converged to the eigenvalues of A. If we use the QR method with double implicit shift to A with the above tolerance, then after 9 iterations we obtain a real Schur form $Q^T AQ = R$. Next consider the following matrix A_1, which we get by perturbing the elements of A:

$$A_1 = \begin{bmatrix} 4.0022 & -4.9949 & 0.0063 & 3.0095 & -1.9959 & 1.0072 \\ 0.0068 & -3.9940 & -2.9956 & -4.9915 & -0.9942 & 2.0080 \\ 5.0091 & 4.0082 & -2.9918 & 0.0029 & 5.0088 & -2.9929 \\ 3.0025 & 0.0076 & 5.0069 & 4.0054 & 2.0044 & -0.9926 \\ -5.9914 & 1.0046 & 3.0070 & -3.9949 & 1.0073 & -1.9998 \\ 2.0047 & -2.9905 & -0.9901 & 1.0010 & 5.0087 & -2.9911 \end{bmatrix}$$

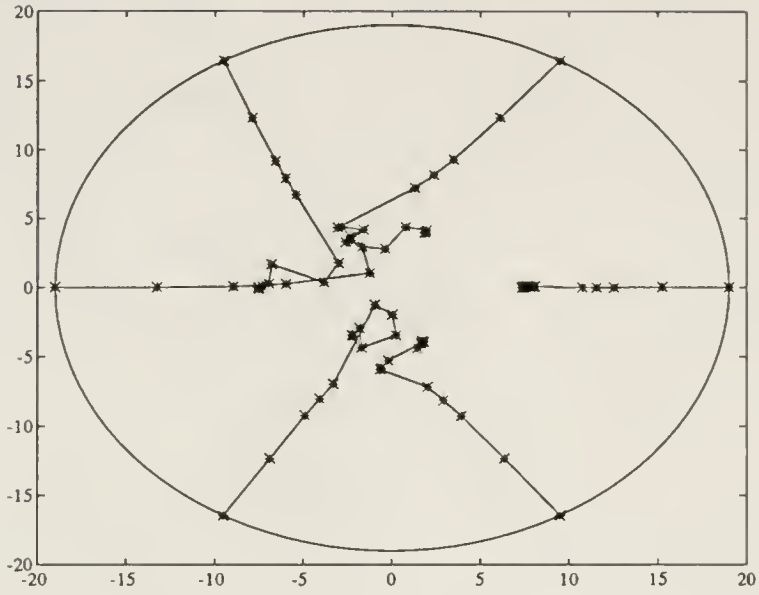

Figure 2.1. The path of convergence of the eigenvalues of the matrix A.

If we use Algorithm 2.6.1 to A_1 with $U_0 = U, S_0 = S$, the tolerance, and the coalescing tolerance are as above, then after 4 iterations we get a block Schur decomposition $P^H A_1 P = \Omega$, whereas if we apply the QR method with double implicit shift to $Q^T A_1 Q$ with the same tolerance, then after 8 iterations we obtain a real Schur form $V^T(Q^T A_1 Q)V = \Gamma$.

Example 2.6.5 If Algorithm 2.6.1 is applied to the matrices

$$B = \begin{bmatrix} 0 & 0 & 0 & -10 \\ 1 & 0 & 0 & 18 \\ 0 & 1 & 0 & -15 \\ 0 & 0 & 1 & 6 \end{bmatrix}, \; C = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & -6 \\ 0 & 0 & 1 & 4 \end{bmatrix},$$

$$D = \begin{bmatrix} 3 & 6 & -2 & -2 \\ -4 & -9 & 4 & 2 \\ -4 & -10 & 5 & 2 \\ -4 & -8 & 4 & 1 \end{bmatrix}, \text{ and } E = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

with diagonal matrices, Λ = diag(19, 3 + 16i, −13, 3 − 16i), Θ = diag(7, 2 + 5i, −3, 2 − 5i), Σ = diag(28, −2 + 30i, −32, −2 − 30i), and Ω = diag(1, 0.5 + 0.8660i, −0.5 + 0.8660i, −1, −0.5 − 0.8660i, 0.5 − 0.8660i) as U_0, where the elements of Λ, Θ, Σ, and Ω are the uniformly distributed points on circles, which include all Gerschgorin disks for B, C, D, and E respectively, $S_0 = I$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 16, 112, 12, and 11 iterations we obtain block Schur decompositions of B, C, D, and E respectively. If we use the QR method with double implicit shift, then after 6, 14, and 1 iterations we obtain real Schur forms of B, C, and D respectively, whereas E diverges.

Example 2.6.6 If we apply Algorithm 2.6.1 to

$$A = X\text{diag}(\ 1,\ 1,\ 1,\ 1,\ 2,\ 2,\ 2,\ 3,\ 3,\ 0,\ -1,\ -1,\ -1,\ -1,\ -1,\ -5,\ -5,\ 4,\ 5,\ -4\ )X^{-1},$$

where $X = P\text{diag}(1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10,\ -1,\ -2,\ -3,\ -4,\ -5,\ -6,\ -7,\ -8,\ -9,\ -10\ )Q$, P, and Q are orthogonal matrices with U_0 = diag(43.6939, 41.5811+ 13.3395i, 35.4496+25.3733i, 25.8996+34.9233i, 13.8658+41.0548i, 0.5263+43.1676i, −12.8132+41.0548i, −24.8470+34.9233i, −34.3970+25.3733i, −40.5285+13.3395i, −42.6413, −40.5285−13.3395i, −34.3970−25.3733i, −24.8470−34.9233i, −12.8132− 41.0548i, 0.5263−43.1676i, 13.8658−41.0548i, 25.8996−34.9233i, 35.4496−25.3733i, 41.5811−13.3395i), where the elements of U_0 are the uniformly distributed points on a circle, which includes all Gerschgorin disks for A, $S_0 = I$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 29 iterations we obtain a block Schur decomposition $S^H AS = U$, where $U = D + N$, D = diag(−1, −1, −1, −1, −1, 1, 1, 1, 1, 2, 2, 2, 3, 3, −5, −5, 0, 4, −4, 5), and N is the strictly upper triangular part of U. If we use the QR method with double implicit shift to A, then after 18 iterations we get a real Schur form $Q^T AQ = R$. Next consider the perturbed matrix $A_1 = A + P/1000$, where P is the matrix from Example 2.1.3. If we apply Algorithm 2.6.1 to A_1 with $U_0 = U$, $S_0 = S$,

the tolerance, and the coalescing tolerance are the same, then after 4 iterations we obtain a block Schur form $P^H A_1 P = \Omega$, whereas if we apply the QR method with double implicit shift to $Q^T A_1 Q$, then after 19 iterations we obtain a real Schur form $V^T(Q^T A_1 Q)V = \Gamma$.

Example 2.6.7 If Algorithm 2.6.1 is applied to the matrix C in Example 2.1.3 with $U_0 = \text{diag}(139.50, 132.6969+42.9537i, 112.9537+81.7022i, 82.2022+112.4537i, 43.4537 +132.1969i, 0.50+139.00i, -42.4537+132.1969i, -81.2022+112.4537i, -111.9537+ 81.7022i, -131.6969+42.9537i, -138.50, -131.6969-42.9537i, -111.9537-81.7022i, -81.2022-112.4537i, -42.4537-132.1969i, 0.50-139.0i, 43.4537-132.1969i, 82.2022 -112.4537i, 112.9537 - 81.7022i, 132.6969 - 42.9537i$), where the elements of U_0 are the uniformly distributed points on a circle, which includes all Gerschgorin disks for C, $S_0 = I$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 26 iterations we obtain a block Schur form $S^H C S = U$. Figure 2.2 shows how the initial eigenvalues of U_0 converged to the eigenvalues of C. If we use the QR method with double implicit shift to C, then after 27 iterations we obtain a real Schur form $Q^T C Q = R$. Next consider the perturbed matrix $C_1 = C + P/1000$, where P is the matrix from Example 2.1.3. If we apply Algorithm 2.6.1 to C_1 with $U_0 = U$, $S_0 = S$, the tolerance, and the coalescing tolerance are the same, then after 2 iterations we obtain a block Schur form $P^H C_1 P = \Omega$, whereas if we apply the QR method with double implicit shift to $Q^T C_1 Q$, then after 27 iterations we obtain a real Schur form $V^T(Q^T C_1 Q)V = \Gamma$.

In the following table for different values of n we give the number of iterations required to obtain block Schur decompositions of A by Algorithm 2.6.1 with diagonal matrices as U_0, where the elements of U_0 are the uniformly distributed points on circles, which include all Gerschgorin disks for A, $S_0 = I$, and real Schur forms of A by the QR method with double implicit shift. Here $A = (a_{ij})$ are random matrices,

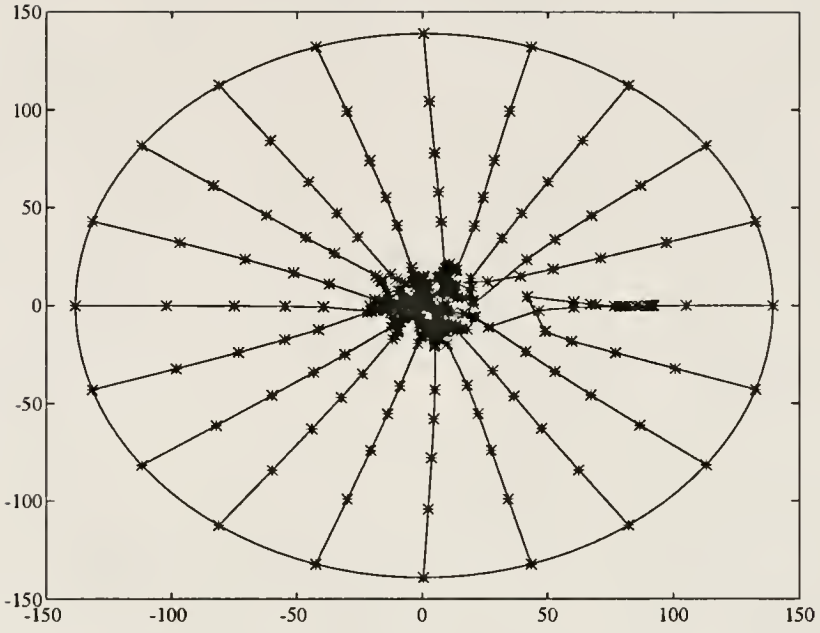

Figure 2.2. The path of convergence of the eigenvalues of C.

where a_{ij} are real numbers and $0 < a_{ij} < 1$, the tolerance $tol = 10^{-6}$, and the coalescing tolerance $tol1 = 10^{-4}$:

size	number of iterations for our algorithm	number of iterations for the QR algorithm with double shift
20	25	27
40	44	58
60	64	79
80	75	110
100	80	134
120	93	178
140	103	193

Algorithm 2.6.1 can be applied to all matrices A with $S_0 = I$, and diagonal matrices as U_0, where the elements of U_0 are the uniformly distributed points on circles, which include all Gerschgorin disks for A. Usually the stepsize in the first iteration becomes 1. If we use this stepsize, then after one iteration all diagonal elements of the resulting upper triangular matrix become real, and this creates problem in the

convergence. To prevent it, we take one fourth of the stepsize from iteration 1 to iteration 2 or 4. If the size of the matrix is very large, then we may have to take one fourth of the stepsize from iteration 1 to iteration 6 or 10. The reason of doing this is the following. If all diagonal entries of the upper triangular matrix at the beginning become real, then the approximate eigenvalues are clustered on a segment of the real line. When two elements are close to each other, but not within the coalescing tolerance, then in the updating process we are dividing by a small number (which is their difference). If this happens in almost all iterations at the beginning, which is the case when the size of a matrix is greater than 10, then it brings instability in the updating process. In fact for that reason the real version of Algorithm 2.6.1 does not work very well, when a diagonal matrix formed from the diagonal elements of the given matrix is used as the starting upper triangular matrix.

2.7 Parallel Processing in Eigencomputations

We already discussed the derivation of five algorithms in the previous sections to find all eigenvalues and corresponding eigenvectors of a matrix. To implement each algorithm, we need either a matrix of approximate eigenvectors, or all approximate eigenvalues, or both. But except for the last one, we have not been able to come up with ways how to choose either a matrix of approximate eigenvectors, or all approximate eigenvalues for the starting guess. So in this section, we will only discuss the implementation of the last algorithm in a parallel computer.

Most of the algorithms, which are available to find all eigenvalues of an unsymmetric matrix are serial algorithms. For example, consider the famous QR algorithm, which is used to find all eigenvalues of an $n \times n$ unsymmetric matrix A. To make it computationally feasible first, A is reduced to a Hessenberg form H, and then double implicit shift (Francis QR Step) is used in each iteration to obtain a real Schur

form. The algorithm, which reduces an $n \times n$ matrix A to a Hessenberg form H, and the algorithm which uses double implicit shift to H are both serial algorithms.

The motivation behind the derivation of various algorithms in this chapter is to find all eigenvalues of an unsymmetric matrix by making use of a sensitivity result for eigenvalues and eigenvectors. Then try to determine whether an algorithm can be implemented in a parallel computer. We already mentioned that except for the last algorithm, all other algorithms described in this chapter have some shortcomings. So we will give a sketch, how and where to modify the last algorithm such that it can be implemented in a parallel computer. For clarity, we will assume that $n = rp$, where p is the number of processors. We will also use the notation Proc(i) to denote the ith processor.

In Step 1, with p processors, we can accelerate the merging of diagonal blocks in the following way. For a fixed i, $1 \leq i < NB$, and $NB - i < p$, we can find $\alpha_j = \min\{\ |\sigma - \omega|\ :\ \sigma \in \lambda(U_{ii})$, and $\omega \in \lambda(U_{jj})\ \}$, $j = i+1, i+2, \ldots, NB$, in separate processors. If $\alpha_j < tol1$, then we can merge U_{jj} with U_{ii} in the increasing order of j. If $NB - i > p$, then we have to use some processors more than once. Proc(ξ) will compute α_l, where $l \in \xi + i : p : NB$. Parallel processors can also be used to arrange diagonal blocks of U in the decreasing order of sizes. Let $[x]$ denote the smallest integer such that $x \leq [x] < x + 1$. If U has k blocks along the diagonal, then not more than $\dfrac{k(k-1)}{2}$ swaps among the diagonal blocks are necessary to arrange them in the decreasing order of sizes. In a single processor, we have to execute $\dfrac{k(k-1)}{2}$ times to finish $\dfrac{k(k-1)}{2}$ swaps. But in a parallel computer with p processors, we can use m processors, where $m = p$ if $p \leq \left[\dfrac{k}{2}\right]$, and $m = \left[\dfrac{k}{2}\right]$ if $\left[\dfrac{k}{2}\right] < p$ to finish $\dfrac{k(k-1)}{2}$ swaps by executing at most $\left[\dfrac{k(k-1)}{2m}\right] + 1$ swaps in each processor. For example, if the dim(NS) is five (say $NS = [\ a_1\ a_2\ a_3\ a_4\ a_5\ ]^T$ with $a_1 < a_2 < a_3 < a_4 < a_5$) and

there are two processors, then we can swap the elements of NS in the following way:

$$\left[\begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{matrix}\right] \left[\begin{matrix} \\ \\ \\ \\ a_5 \end{matrix}\right] \rightarrow \left[\begin{matrix} a_2 \\ a_1 \\ a_4 \\ a_3 \end{matrix}\right] \left[\begin{matrix} \\ \\ \\ \\ a_5 \end{matrix}\right] \rightarrow \left[\begin{matrix} a_2 \\ a_4 \\ a_1 \\ a_5 \end{matrix}\right] \left[\begin{matrix} \\ \\ \\ \\ a_3 \end{matrix}\right] \rightarrow \left[\begin{matrix} a_4 \\ a_2 \\ a_5 \\ a_1 \end{matrix}\right] \left[\begin{matrix} \\ \\ \\ \\ a_3 \end{matrix}\right] \rightarrow \left[\begin{matrix} a_4 \\ a_5 \\ a_2 \\ a_3 \end{matrix}\right] \left[\begin{matrix} \\ \\ \\ \\ a_1 \end{matrix}\right] \rightarrow \left[\begin{matrix} a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \end{matrix}\right]$$

Here we use one processor to swap blocks enclosed by a brace on the right, and the other processor to swap blocks enclosed by a parenthesis on the right. Each processor does 5 comparisons, and 5 swaps.

In Step 2, to compute the strictly lower triangular matrix $G(S,U)$, we can use parallel processors. Suppose $A = [A_1, \ldots, A_p]$, $S = [S_1, \ldots, S_p]$, and $U = [U_1, \ldots, U_p]$ are column partitionings of A, S, and U respectively, where each block column has width r. To compute $B = AS - SU$, first we find $D = SU$, and then find $B = AS - D$. Since $U_j = \left[U_{1j}^T, \ldots, U_{jj}^T, 0, \ldots, 0 \right]^T$, $U_{ij} \in \mathbb{C}^{r \times r}$, so $D_k = SU_k = \sum_{j=1}^{k} S_j U_{jk}$, $k = 1, \ldots, p$. Thus each D_k can be computed in a separate processor. Since U is upper triangular, so D_p involves much more work than D_1. To load balance, we assign Proc(ξ) the computation of

$$D(:, \xi : p : n) = SU(:, \xi : p : n) = \sum_{l=1}^{p} S(:, l : p : n) U(l : p : n, \xi : p : n),$$

as suggested in [Gol90, page 291]. Next with $S_j = \left[S_{1j}^T, \ldots, S_{pj}^T \right]^T$, $S_{ij} \in \mathbb{C}^{r \times r}$,

$$B_k = AS_k - D_k = \sum_{j=1}^{p} A_j S_{jk} - D_k, \quad k = 1, \ldots, p.$$

So p processors can be used to compute B, where D can be overwritten by B. As we mentioned in Section 2.5, to compute G_{ij} we need entries below G_{ij} on the jth block column; namely $G_{i+1j}, \ldots, G_{NBj}$, and entries on the left of G_{ij} on the ith block row; namely $G_{i1}, \ldots, G_{ij-1}$. To load balance, we will use the following two strategies to compute the elements of $G(S,U)$; one for $p \geq \left[\frac{NB}{2} \right] + 1$, and the other for $p < \left[\frac{NB}{2} \right] + 1$.

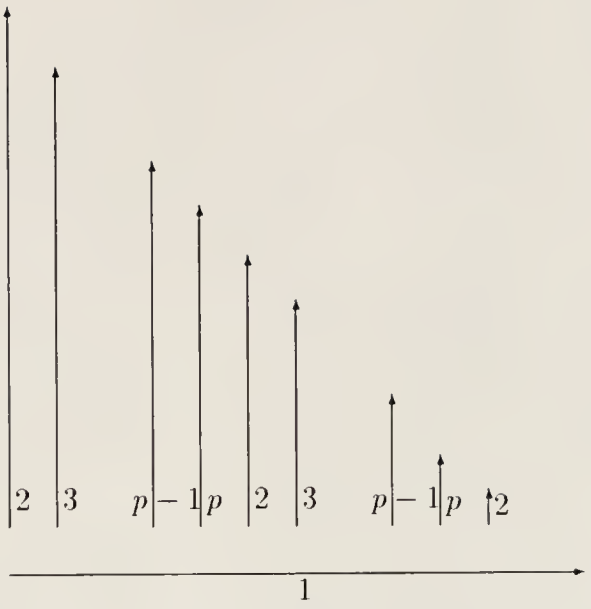

Figure 2.3. The direction, in which the elements of $G(S,U)$ are computed, when the number of processors are greater than half of the total blocks.

Suppose $p \geq \left[\frac{NB}{2}\right] + 1$. Then we use Proc(1) to compute G_{NBj}, $j = 1,\ldots,$ $NB-1$ in the increasing order of the column index j, and the other $p-1$ processors to compute the remaining elements of $G(S,U)$ in the decreasing order of the row index i, as we explain below. We use Proc(2) to compute G_{i1}, $i = NB-1,\ldots,2$, Proc(3) to compute G_{i2}, $i = NB-1,\ldots,3$. Continuing the assignment in this way, we use Proc(p) to compute G_{ip-1}, $i = NB-1,\ldots,p$. If $NB-1 > p$, then we will continue this ascending order of assignments to those processors except Proc(1). That means, Proc(2) to compute G_{ip}, $i = NB-1,\ldots,p+1$, Proc(3) to compute G_{ip+1}, $i = NB-1,\ldots,p+2$, and in this order if necessary Proc(p) to compute G_{i2p-2}, $i = NB-1,\ldots,2p-1$. Pictorially it can be presented by arrow heads, where the elements are computed towards the arrow heads as in Figure 2.3.

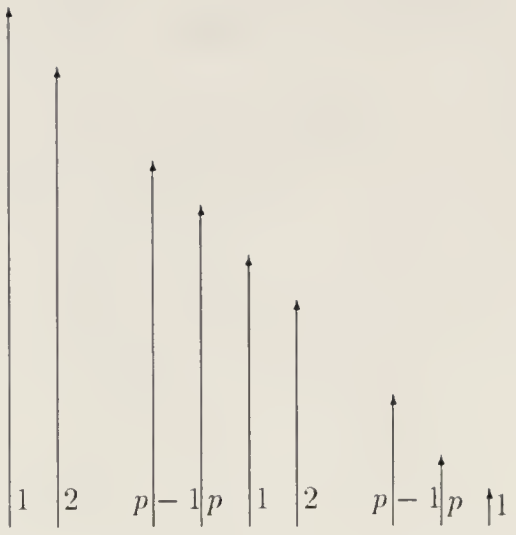

Figure 2.4. The direction, in which the elements of $G(S,U)$ are computed, when the number of processors are less than half of the total blocks.

Next, suppose $p < \left[\frac{NB}{2}\right] + 1$. This time we will assign each processor to compute the elements in the decreasing order of the row index i. Proc(1) to compute G_{i1}, $i = NB,\ldots,2$. Proc(2) to compute G_{i2}, $i = NB,\ldots,3$. Assigning the columns in this order, we use Proc(p) to compute G_{ip}, $i = NB,\ldots,p+1$. Next we will repeat the order of assignment as in the first strategy. Proc(1) to compute G_{ip+1}, $i = NB,\ldots,p+2$, Proc(2) to compute G_{ip+2}, $i = NB,\ldots,p+3$, and in this order Proc(p) to compute G_{i2p}, $i = NB,\ldots,2p+1$. If necessary, we will repeat it. Pictorially it can be shown by arrow heads, where the elements are computed towards the arrow heads as in Figure 2.4.

In both strategies, we can not start all processors simultaneously. In the first strategy, before Proc(ξ), $\xi = 2,\ldots,p$, computes an element G_{ij}, we need to make sure that G_{il}, $l = 1,\ldots,j-1$ are already computed, and if $i = NB-1$, then make sure that Proc(1) already computed G_{NBj}. The number of flops required by the serial

algorithm to compute $G(S,U)$ is a complicated function of the block sizes of U. If all blocks are 1×1 matrices, then the algorithm requires $\frac{5}{3}n^3$ flops.

In Step 3, to evaluate $f(t)$ for different values of t, we can use parallel processors. To compute the lower triangular matrix L, where $L_{ij} = tG_{ij}$ for $1 \leq j < i \leq NB$, we can use one of the strategies discussed earlier to compute $G(S,U)$, depending on the number of processors p. Assign Proc(ξ), $1 \leq \xi \leq p$, to take $L_{jj} = I$, where $j = p\alpha + \xi$, $\alpha = 0, 1, \ldots$, and $j \leq NB$. We can use the ideas discussed in the analysis of Step 2, to find matrix products. Next we will give a sketch to find the inverse of a matrix in a distributed memory multiprocessor, where processors are set up in a ring. An efficient way to find the inverse of a matrix A is to find the LU factorization of A with partial pivoting first. Let $piv(1:n-1) = 0$ be a zero vector. Consider the following node program, which is due to G. H. Golub and C. F. Van Loan, where ξ is the identity of the ξth processor:

$B = A(1:n, \xi:p:n)$; $pivot = piv(\xi:p:n-1)$

$j = 1$; $q = 1$; $col = \xi:p:n$; $L = \text{length}(col)$

while $q \leq L$

if $j = col(q) \wedge j < n$

{ Find the permutation index $piv(j)$, and Gauss vector $A(j+1:n, j)$. }

Determine k with $j \leq k \leq n$, so $|B(k,q)| = ||B(j:n,q)||_\infty$

$pivot(q) = k$; $B(j, 1:L) \leftrightarrow B(k, 1:L)$

$B(j+1:n, q) = B(j+1:n, q)/B(j,q)$

Send $pivot(q)$, $B(j+1:n, q)$ to processors on the right.

{ Update local columns. }

if $q < L$

$B(j+1:n, q+1:L) = B(j+1:n, q+1:L)$

$-B(j+1:n, q)B(j, q+1:L)$

end

$j = j + 1;\ q = q + 1$

else

Receive $piv(j)$, and $A(j+1:n, j)$ from the left, and if the processor on the right is not the processor which computed $A(j+1:n, j)$, and j is less than the last column index of the processor on the right, then send $piv(j)$, and $A(j+1:n, j)$ to the right.

{ Update local columns. }

$B(j, 1:L) \leftrightarrow B(piv(j), 1:L)$

$B(j+1:n, q:L) = B(j+1:n, q:L) - A(j+1:n, j)B(j, q:L)$

$j = j + 1$

end

end

After each processor in a p-processor ring finishes the execution of the above code, we will get the LU factorization of A; Proc(ξ) will house $A(1:n, \xi:p:n)$ in a local array B, and $piv(\xi:p:n-1)$ in a local vector *pivot*. The above LU factorization of A is of the form $PA = LR$. After simplifying, we get $A^{-1} = R^{-1}L^{-1}P$, and therefore to obtain A^{-1}, we need to determine L^{-1}, R^{-1}, and $R^{-1}L^{-1}$. First, we will discuss how to find the inverse B of a lower triangular matrix L, where L is overwritten by B. Consider the following node program:

$D = [L(k:n, k)],\ k = \xi:p:n;\ j = 1$

$q = 1;\ col = \xi:p:n;\ N = \text{length}(col)$

while $j \leq n$

if $j = col(q)$

if $j > 1$

Send $D(j:n, q)$ to processors on the left.

end

Update local columns. }

if $q > 1$

$D(j, 1 : q-1) = D(j, 1 : q-1)/D(j, q)$

if $j < n$

$D(j+1 : n, 1 : q-1) = D(j+1 : n, 1 : q-1)$

$- D(j+1 : n, q)D(j, 1 : q-1)$

end

end

$D(j, q) = 1/D(j, q)$

if $j < n$

$D(j+1 : n, q) = -D(j+1 : n, q)D(j, q)$

end

$j = j + 1$

else

if $j > col(1)$

Receive $L(j : n, j)$ from the right, and if the processor on the left is not the processor which houses $L(j : n, j)$, and j is greater than the first column index of the processor on the left, then send $L(j : n, j)$ to the left.

{ Update local columns. }

$D(j, 1 : q) = D(j, 1 : q)/L(j, j)$

if $j < n$

$D(j+1 : n, 1 : q) = D(j+1 : n, 1 : q) - L(j+1 : n, j)D(j, 1 : q)$

end

$j = j + 1$

```
            if col(q) + p = j
                q = q + 1
            end
        else
            j = j + 1
        end
    end
end
```

If each node in a p-processor ring executes the above code, then we will get the inverse of L, and upon completion Proc(ξ) will house $L(k:n,k)$ for $k = \xi : p : n$ in a local array D. Our next move is to find the inverse B of an upper triangular matrix R, where R is overwritten with B. A node program can be structured as follows:

```
D = [R(1 : k, k)], k = ξ : p : n; j = n
col = ξ : p : n; N = length(col); q = N
while j ≥ 1
    if j = col(q)
        if j < n
            Send D(1 : j, q) to processors on the right.
        end
        { Update local columns. }
        if q < N
            D(j, q + 1 : N) = D(j, q + 1 : N)/D(j, q)
            if j > 1
                D(1 : j − 1, q + 1 : N) = D(1 : j − 1, q + 1 : N)
                                        − D(1 : j − 1, q)D(j, q + 1 : N)
            end
```

```
        end
        D(j,q) = 1/D(j,q)
        if j > 1
            D(1 : j − 1, q) = −D(1 : j − 1, q)D(j, q)
        end
        j = j − 1
    else
        if j < col(N)
            Receive R(1 : j, j) from the left, and if the processor on the right is
            not the processor which houses R(1 : j, j), and j is less than the last
            column index of the processor on the right, then send R(1 : j, j) to the
            right.
            { Update local columns. }
            D(j, q : N) = D(j, q : N)/R(j, j)
            if j > 1
                D(1 : j − 1, q : N) = D(1 : j − 1, q : N) − R(1 : j − 1, j)D(j, q : N)
            end
            j = j − 1
            if col(q) − p = j
                q = q − 1
            end
        else
            j = j − 1
        end
    end
end
```

If each processor in a p-processor ring executes the above node program, then upon completion Proc(ξ) will house $R(1:k,k)$ for $k = \xi : p : n$ in a local array D. In the LU factorization of A, R can be stored in the upper triangular part of A, and the unit lower triangular matrix L can be stored in the strictly lower triangular part of A. Then we can use the above code to find the inverse of the upper triangular part of A, and after a slight modification of the node program, which finds the inverse of a lower triangular matrix, we can use it to find the inverse of the strictly lower triangular part of A. Since $A^{-1} = R^{-1}L^{-1}P$, so our next goal is to create a node program to find the product $R^{-1}L^{-1}$. Consider the following node program:

$B = A(1:n, \xi : p : n)$; $j = 1$; $col = \xi : p : n$; $L = \text{length}(col)$; $q = 1$

while $j \leq n$

if $j = col(q)$

if $j > 1$

Send $B(1:j,q)$ to processors on the left.

end

{ Update local columns. }

if $q > 1$

$$B(1:j,1:q-1) = \begin{bmatrix} B(1:j-1,1:q-1) \\ 0 \end{bmatrix} + B(1:j,q)B(j,1:q-1)$$

{ 0 is a $q-1$ dimensional row zero vector. }

end

$j = j + 1$

else

if $j > col(1)$

Receive $A(1:j,j)$ from the right, and if the processor on the left is not the processor which houses $A(1:j,j)$, and j is greater than the first column index of the processor on the left, then send $A(1:j,j)$ to

the left.

{ Update local columns. }

$B(1:j,1:q) = \begin{bmatrix} B(1:j-1,1:q) \\ 0 \end{bmatrix} + A(1:j,j)B(j,1:q)$

$j = j+1$

if $col(q) + p = j$

$q = q+1$

end

else

$j = j+1$

end

end

end

After each processor in a p-processor ring executes the above code, we will obtain the product $R^{-1}L^{-1}$ which overwrites A, and upon completion Proc(ξ) will house $A(1:n,\xi:p:n)$ in a local array B. Since A is overwritten by $R^{-1}L^{-1}$, and the permutation matrix P is encoded in the $(n-1)$ dimensional vector piv, so $A^{-1} = (R^{-1}L^{-1})P$ can be computed in the following way:

for $k = n-1:-1:1$

if $k \neq \text{piv}(k)$

$A(1:n,k) \leftrightarrow A(1:n,\text{piv}(k))$

end

end

Next, if we store the upper triangular matrix U in the upper triangular part of a $n \times n$ matrix B, and the strictly lower triangular matrix $G(S,U)$ in the strictly lower triangular part of B, then the code, which is used to compute the product $R^{-1}L^{-1}$ can be applied to B to find the product $UG(S,U)$. Similarly, it is possible to find the

product $G(S,U)U$ in a more efficient way. After computing $UG(S,U)$, S^{-1}, $S^{-1}AS$, and $G(S,U)U$, we can take advantage of parallel processors to compute $U(t) = U + t\,(UG(S,U) + S^{-1}AS - U - G(S,U)U)$. Next after $Z(t) = A - S(t)U(t)S(t)^{-1}$ is calculated, we can find $f(t) = \sqrt{\text{trace}(Z(t)^H Z(t))}$ in the following way. Let $x(1:p) = 0$ be a zero vector. Let $Z = Z(t)$, and $r = \dfrac{n}{p}$. To compute $Z^H Z$ we use the following procedure:

$B = Z(1:n, 1+(\xi-1)r : \xi r);\ y = x(\xi)$

for $i = 1 : r$

$y = y + [B(1:n,i)]^H B(1:n,i)$

end

Here each processor can execute the above code simultaneously, and upon completion Proc(ξ) houses $x(\xi)$ in a local variable y. Then $f(t) = \sqrt{x_1 + \ldots + x_p}$.

In Step 4, if $NB - m < p$, then we can not use all processors to find il, and iu. Suppose $NB - m > p$. Using a similar technique, as described in the analysis of Step 1, we can get an array k such that $|\mu_k(i)| \le |\mu_k(i+1)|$ holds. Let

$$r_i = \frac{\mu_{k(i+1)} - \mu_{k(i)}}{d_{k(i)} - d_{k(i+1)}}.$$

To determine $|r_{i_0}| = \min\{\ |r_i|\ :\ i = 1, \ldots, NB - m - 1,\ \text{Re}\,(r_i) > 0\ \&\ |r_i| < 1\ \}$, we can use p processors as follows. Let $s = \left\lceil \dfrac{NB-m}{p} \right\rceil$. Define

$$\begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_{p-1} \\ B_p \end{bmatrix} = \begin{bmatrix} \mu_{k(1:s+1)} \\ \mu_{k(s+1:2s+1)} \\ \vdots \\ \mu_{k((p-2)s+1:(p-1)s+1)} \\ \mu_{k((p-1)s+1:NB-m)} \end{bmatrix}, \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_{p-1} \\ C_p \end{bmatrix} = \begin{bmatrix} d_{k(1:s+1)} \\ d_{k(s+1:2s+1)} \\ \vdots \\ d_{k((p-2)s+1:(p-1)s+1)} \\ d_{k((p-1)s+1:NB-m)} \end{bmatrix},$$

$l = [\ 1, s+1, 2s+1, \ldots, (p-1)s+1\ ]^T$, $u(1:p) = 0$, and $v(1:p) = 0$. Next consider the following procedure:

$f = B_\xi;\ g = C_\xi;\ \beta = u(\xi);\ \delta = v(\xi)$

$\nu = l(\xi);\ L = \text{length}(f);\ r(1:L-1) = 0$

for $i = 1 : L-1$

$$r_i = \frac{f_{i+1} - f_i}{g_i - g_{i+1}}$$

end

$\beta = 1;\ j = 0$

for $i = 1 : L-1$

if $\text{Re}(r_i) > 0 \wedge |r_i| < \beta$

$\beta = |r_i|;\ j = i$

end

end

if $j = 0$

$\delta = 0$

else

$\delta = \nu - 1 + j$

end

If each processor executes the above code, then upon completion Proc(ξ) houses $u(\xi) = \beta$, and $v(\xi) = \delta$. Next the desired minimum is $u(\xi_0) = \min\{\ u(\xi),\ \xi = 1, \ldots, p\ \}$, and if the corresponding $v(\xi_0) \neq 0$, then define $il = t - 1 + \min\{k(v(\xi_0)), k(v(\xi_0)+1)\}$, and $iu = t - 1 + \max\{k(v(\xi_0)), k(v(\xi_0)+1)\}$.

In Step 5, we can use parallel processors, as described in the analysis of Step 2, and Step 3 to obtain U^{new}, S^{new}, and $Z = S^{new}U^{new}\,(S^{new})^{-1}$.

In Step 6, if $m \leq p$, then we can assign a separate processor to implement the code to each diagonal block U_{jj}, $1 \leq j \leq m$. If $m > p$, then assign Proc(ξ) to implement the code to diagonal blocks U_{jj}, where $j \in \xi : p : m$. Thus we have:

$col = \xi : p : m;\ L = \text{length}(col)$

for $q = 1 : L$

$\alpha = col(q)$; Implement Step 6 to $U_{\alpha\alpha}$

end

In Step 8, we can use parallel processors to find a QR factorization of S. Here we will discuss how to find a QR factorization of a matrix A in a distributed memory multiprocessor, where processors are interconnected in a ring. Let $b(1 : n-1) = 0$ be a zero vector. Consider the following procedure, which is due to G. H. Golub and C. F. Van Loan:

$B = A(1 : n, \xi : p : n)$; $c = b(\xi : p : n-1)$

$j = 1$; $q = 1$; $col = \xi : p : n$; $L = \text{length}(col)$

while $q \leq L$

if $j = col(q) \wedge j < n$

{ Find Householder vector $A(j+1 : n, j)$. }

$\alpha = ||B(j : n, q)||_2$

if $\alpha = 0$

$B(j : n, q) = 0$; **quit**

else

if $B(j, q) \neq 0$

$\beta = \alpha B(j, q)/|B(j, q)|$

else

$\beta = \alpha$

end

$\beta = B(j, q) + \beta$; $B(j+1 : n, q) = B(j+1 : n, q)/\beta$; $B(j, q) = -\alpha$

end

$$v = \begin{bmatrix} 1 \\ B(j+1 : n, q) \end{bmatrix}; \; c(q) = -\frac{2}{v^H v}$$

Send $c(q)$, and $B(j+1 : n, q)$ to processors on the right.

{ Update local columns. }

if $q < L$

$B(j:n, q+1:L) = B(j:n, q+1:L) + c(q)vv^H B(j:n, q+1:L)$

end

$j = j+1;\ q = q+1$

else

Receive $b(j)$, and $A(j+1:n, j)$ from the left, and if the processor on the right is not the processor which formed $A(j+1:n, j)$, and j is less than the last column index of the processor on the right, then send $b(j)$, and $A(j+1:n, j)$ to the right.

{ Update local columns. }

$$v = \begin{bmatrix} 1 \\ A(j+1:n, j) \end{bmatrix}$$

$B(j:n, q:L) = B(j:n, q:L) + b(j)vv^H B(j:n, q:L);\ j = j+1$

end

end

After each processor in a p-processor ring executes the above code, we will get a QR factorization of A, and Proc(ξ) will house $A(1:n, \xi:p:n)$ in a local array B, and $b(\xi:p:n-1)$ in a local vector c. The upper triangular part of A is overwritten by the upper triangular part of R, and components $j+1:n$ of the jth Householder vector are stored in $A(j+1:n, j)$, $j < n$. Here we need an explicit form of the unitary matrix Q. We find Q using the following procedure, which uses Householder vectors $A(j+1:n, j)$, where $j < n$, and the vector b to form Q explicitly, and zeros subdiagonal elements of A. Let $Q = I_n$:

$B = A(1:n, \xi:p:n);\ I = Q(1:n, \xi:p:n);\ c = b(\xi:p:n-1)$

$j = n-1;\ col = \xi:p:n;\ L = \text{length}(col);\ q = L$

while $j \geq 1$

if $j = col(q)$

Send $c(q)$, and $B(j+1:n,q)$ to procesors on the left.

{ Update local columns. }

$$v = \begin{bmatrix} 1 \\ B(j+1:n,q) \end{bmatrix};\ B(j+1:n,q) = 0$$

$$I(j:n,q:L) = I(j:n,q:L) + c(q)vv^H I(j:n,q:L);\ j = j-1$$

else

Receive $b(j)$, and $A(j+1:n,j)$ from the right, and if the processor on the left is not the processor which houses $A(j+1:n,j)$, and j is greater than the first column index of the processor on the left, then send $b(j)$, and $A(j+1:n,j)$ to the left.

{ Update local columns. }

$$v = \begin{bmatrix} 1 \\ A(j+1:n,j) \end{bmatrix}$$

$$I(j:n,q:L) = I(j:n,q:L) + b(j)vv^H I(j:n,q:L);\ j = j-1$$

if $col(q) - p = j$

$q = q - 1$

end

end

end

If each processor in a p-processor ring executes the above code, then upon completion Proc(ξ) will house $A(1:k,k)$ for $k = \xi : p : n$ in a local array B, and $Q(1:n,\xi:p:n)$ in a local array I. Next we can use parallel processors, as described in the analysis of Step 2 to find the product $U^{new} = Q^H ZQ$.

Finally in Step 9, we can use multiprocessors, as described in the analysis of Step 3 to evaluate $f(0) = ||A - Z||_F$.

CHAPTER 3
EIGENVALUES OF SYMMETRIC MATRICES

3.1 Diagonalization of a Symmetric Matrix using Armijo's Stepsize

Let A be an $n \times n$ symmetric matrix. Suppose the eigenvalues of A are all distinct. In this section we will find an iterative method to compute a diagonalization $A = Q\Lambda Q^T$, where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, and Q is an orthogonal matrix such that the diagonal elements of Λ are the eigenvalues of A, and the columns of Q are eigenvectors of A. Consider the following algorithm to compute the eigenvalues, and corresponding eigenvectors of a matrix A. Detail is given in [Hag88, page 304 – 14], or in Chapter 1:

$$\begin{aligned} \lambda_j^{new} &= \left(y_j^{old}\right)^T A x_j^{old} && \text{for } j = 1 \text{ to } n, \\ x_j^{new} &= x_j^{old} + \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{\left(y_i^{old}\right)^T A x_j^{old}}{\lambda_j^{new} - \lambda_i^{new}} x_i^{old} && \text{for } j = 1 \text{ to } n, \\ Y^{new} &= \left(X^{new}\right)^{-1}. \end{aligned} \tag{3.1}$$

Algorithm in (3.1) can be rewritten in the following way:

$$\begin{aligned} \Lambda^{new} &= \text{Diag}\left(\left(X^{old}\right)^{-1} A X^{old}\right), \\ X^{new} &= X^{old}\left(I + F\left(X^{old}, \Lambda^{new}\right)\right), \end{aligned} \tag{3.2}$$

where $\text{Diag}(M) = \text{diag}(m_{11}, m_{22}, \ldots, m_{nn})$, and

$$f_{ij}\left(X^{old}, \Lambda^{new}\right) = \begin{cases} 0 & \text{for } i = j, \\ \dfrac{\left(\left(X^{old}\right)^{-1} A X^{old}\right)_{ij}}{\lambda_{jj}^{new} - \lambda_{ii}^{new}} & \text{for } i \neq j. \end{cases}$$

In (3.2), if we put $Y^{new} = I + F\left(X^{old}, \Lambda^{new}\right)$, and $A^{old} = \left(X^{old}\right)^{-1} A X^{old}$, then it reduces to the form:

$$\begin{aligned} \Lambda^{new} &= \text{Diag}\left(\left(Y^{old}\right)^{-1} A^{old} Y^{old}\right), \\ X^{new} &= X^{old} Y^{new}. \end{aligned}$$

Hence in each iteration, we can update the eigenvalues, and corresponding eigenvectors in the following way:

$$\begin{aligned} A^{new} &= \left(Y^{old}\right)^{-1} A^{old} Y^{old}, \\ \Lambda^{new} &= \text{Diag}\left(A^{new}\right), \\ Y^{new} &= I + F\left(A^{new}\right), \\ X^{new} &= X^{old} Y^{new}, \end{aligned} \tag{3.3}$$

where

$$f_{ij}\left(A^{new}\right) = \begin{cases} 0 & \text{for } i = j, \\ \dfrac{a_{ij}^{new}}{a_{jj}^{new} - a_{ii}^{new}} & \text{for } i \neq j. \end{cases}$$

The updating process in (3.3) does not restore the symmetry of the given matrix A, in the update $A^{new} = \left(Y^{old}\right)^{-1} A^{old} Y^{old}$. In each iteration to restore the symmetry of A, and the orthogonality of X we do the following. Let $Y^{new} = Q^{new} R^{new}$ be a QR factorization of Y^{new}. Now if we take $A^{new} = \left(Q^{old}\right)^{T} A^{old} Q^{old}$, and $X^{new} = X^{old} Q^{new}$, then A^{new} will be symmetric, and X^{new} will be orthogonal. With these modifications, (3.3) becomes:

$$\begin{aligned} A^{new} &= \left(Q^{old}\right)^{T} A^{old} Q^{old}, \\ \Lambda^{new} &= \text{Diag}\left(A^{new}\right), \\ Y^{new} &= I + F\left(A^{new}\right), \\ Y^{new} &= Q^{new} R^{new}, \quad (\text{ QR factorization }) \\ X^{new} &= X^{old} Q^{new}, \end{aligned} \tag{3.4}$$

where

$$f_{ij}\left(A^{new}\right) = \begin{cases} 0 & \text{for } i = j, \\ \dfrac{a_{ij}^{new}}{a_{jj}^{new} - a_{ii}^{new}} & \text{for } i \neq j. \end{cases}$$

If the starting guess matrix of eigenvectors Q_0 is not a good approximation of Q in the factorization $A = Q \Lambda Q^T$, then in the update $Y^{new} = I + F\left(A^{new}\right)$,

the increment $F(A^{new})$ may be too large. However with a large increment, we may likely have instabilitiy in the updating process, and the algorithm may diverge. To restore the convergence of the algorithm, and to have a steady change in the value of Y^{new}, we need a small increment in each iteration. To achieve this, we redefine each iterate, and introduce a small positive parameter. Let s be a positive parameter, and define $Z(s)^{new} = I + sF(A^{new})$. So $Z(0)^{new} = I$, and $Z(1)^{new} = Y^{new}$, the matrix generated by (3.4). Define $f(s) = ||G(s)||_F$, where $G(s) = \text{lotr}\left(\left(Z(s)^{old}\right)^{-1} A^{old} Z(s)^{old}\right)$, lotr$(B)$ is a strictly lower triangular matrix formed from the strictly lower triangular part of B. Suppressing the superscripts in $Z(s)^{new}$, and $G(s)$, we have $Z(s) = I + sF(A)$, and $G(s) = \text{lotr}\,(Z(s)^{-1}AZ(s))$. Hence $f(0) = ||\text{lotr}\,(A)\,||_F$, and $f(1) = ||\text{lotr}\,(Y^{-1}AY)\,||_F$.

With a starting guess X_0, if the algorithm in (3.4) converges, then we must have $f(1) \leq f(0)$. As in Chapter 2, here our goal is to find an s, $0 < s \leq 1$, for which $f(s) \leq f(0)$ holds. To this end, we use Armijo's rule from optimization theory. In Armijo's rule, we determine s in the following way. Evaluate $f(s)$ at $s = 1, \frac{1}{2}, \frac{1}{4}, \cdots$, stopping when

$$f(s) \leq \left(1 - \frac{s}{2}\right) f(0).$$

To use the above inequality, $f(s)$ must satisfy $f'(0) = -f(0)$. Hence our next goal is to evaluate $f'(0)$, when $f(s) = ||G(s)||_F$ with $G(s) = \text{lotr}\,(Z(s)^{-1}AZ(s))$. Before we find $f'(0)$, we will prove the following basic result.

Lemma 3.1.1 *If $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix, and $B \in \mathbb{R}^{n\times n}$ is a skew symmetric matrix, then $AB - BA$ is a symmetric matrix.*

Proof of Lemma 3.1.1 Let A be a symmetric matrix, and B be a skew symmetric matrix. So $A^T = A$, and $B^T = -B$. Now

$$(AB - BA)^T = (AB)^T - (BA)^T \quad .$$

$$
\begin{aligned}
&= B^T A^T - A^T B^T \\
&= (-B)\,A - A\,(-B) \\
&= AB - BA
\end{aligned}
$$

Hence $AB - BA$ is a symmetric matrix. □

With the above definition of $f(s)$, now we are ready to find $f'(0)$, and will show that $f'(0) = -f(0)$.

Theorem 3.1.1 *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Suppose its eigenvalues are all distinct. Define $Z(s) = I + sF(A)$, $G(s) = lotr\left((Z(s))^{-1} AZ(s)\right)$, and $f(s) = ||G(s)||_F$, where*

$$
f_{ij}(A) = \begin{cases} 0 & for \quad i = j, \\ \dfrac{a_{ij}}{a_{jj} - a_{ii}} & for \quad i \neq j, \end{cases}
$$

$lotr(B)$ is a strictly lower triangular matrix formed from the strictly lower triangular part of B. Then $f'(0) = -f(0)$.

Proof of Theorem 3.1.1 Using the result (2.18) of Lemma 2.3.1, we have

$$
f(0)f'(0) = \text{trace}\left(G(0)^T \frac{d}{ds} G(s)|_{s=0}\right). \tag{3.5}
$$

Differentiating $G(s)$ with respect to s, and then evaluating the derivative at $s = 0$, we obtain

$$
\frac{d}{ds} G(s)|_{s=0} = \text{lotr}\left(-Z(0)^{-1} \frac{d}{ds} Z(s)|_{s=0} Z(0)^{-1} AZ(0) + Z(0)^{-1} A \frac{d}{ds} Z(s)|_{s=0}\right). \tag{3.6}
$$

Differentiating $Z(s)$ with respect to s, we have $\dfrac{d}{ds} Z(s) = F$. Hence $\dfrac{d}{ds} Z(s)|_{s=0} = F$. Substituting this value, and the value of $Z(0) = I$ in (3.6), and then simplifying we get

$$
\frac{d}{ds} G(s)|_{s=0} = \text{lotr}\,(AF - FA). \tag{3.7}
$$

Since $G(0) = \text{lotr}\,(Z(0)^{-1}AZ(0)) = \text{lotr}\,(IAI) = \text{lotr}(A)$, so (3.5) becomes

$$f(0)f'(0) = \text{trace}\left((\text{lotr}(A))^T \text{lotr}\,(AF - FA)\right). \tag{3.8}$$

Because A is symmetric, so $(\text{lotr}\,(A))^T = \text{uptr}\,(A)$, where uptr$(A)$ is a strictly upper triangular matrix formed from the strictly upper triangular part of A. Hence (3.8) reduces to

$$f(0)f'(0) = \text{trace}\,(\text{uptr}\,(A)\,\text{lotr}\,(AF - FA)). \tag{3.9}$$

Next, for $i \neq j$, $f_{ij} = \dfrac{a_{ij}}{a_{jj} - a_{ii}}$, and $f_{ji} = \dfrac{a_{ji}}{a_{ii} - a_{jj}}$. Since A is symmetric, so $f_{ij} = -f_{ji}$. That is F is a skew symmetric matrix. Hence by Lemma 3.1.1, $AF - FA$ is symmetric. Now $AF - FA =$

$$\begin{bmatrix}
\displaystyle\sum_{i=2}^{n} \frac{2\,a_{i1}^2}{a_{11} - a_{ii}} & \displaystyle\sum_{\substack{i=1\\ i\neq 2}}^{n} \frac{a_{1i}a_{i2}}{a_{22} - a_{ii}} + \sum_{i=2}^{n} \frac{a_{1i}a_{i2}}{a_{11} - a_{ii}} & \cdots & \displaystyle\sum_{i=1}^{n-1} \frac{a_{1i}a_{in}}{a_{nn} - a_{ii}} + \sum_{i=2}^{n} \frac{a_{1i}a_{in}}{a_{11} - a_{ii}} \\
\displaystyle\sum_{i=2}^{n} \frac{a_{2i}a_{i1}}{a_{11} - a_{ii}} + \sum_{\substack{i=1\\ i\neq 2}}^{n} \frac{a_{2i}a_{i1}}{a_{22} - a_{ii}} & \displaystyle\sum_{\substack{i=1\\ i\neq 2}}^{n} \frac{2\,a_{i2}^2}{a_{22} - a_{ii}} & \cdots & \displaystyle\sum_{i=1}^{n-1} \frac{a_{2i}a_{in}}{a_{nn} - a_{ii}} + \sum_{\substack{i=1\\ i\neq 2}}^{n} \frac{a_{2i}a_{in}}{a_{22} - a_{ii}} \\
\vdots & \vdots & & \vdots \\
\displaystyle\sum_{i=2}^{n} \frac{a_{ni}a_{i1}}{a_{11} - a_{ii}} + \sum_{i=1}^{n-1} \frac{a_{ni}a_{i1}}{a_{nn} - a_{ii}} & \displaystyle\sum_{\substack{i=1\\ i\neq 2}}^{n} \frac{a_{ni}a_{i2}}{a_{22} - a_{ii}} + \sum_{i=1}^{n-1} \frac{a_{ni}a_{i2}}{a_{nn} - a_{ii}} & \cdots & \displaystyle\sum_{i=1}^{n-1} \frac{2\,a_{in}^2}{a_{nn} - a_{ii}}
\end{bmatrix}.$$

Therefore (3.9) gives:

$$f(0)f'(0) =$$

$$\text{trace}\left(\begin{bmatrix}
\displaystyle\sum_{k=2}^{n} a_{1k}\left(\sum_{i=2}^{n} \frac{a_{ki}a_{i1}}{a_{11} - a_{ii}} + \sum_{\substack{i=1\\ i\neq k}}^{n} \frac{a_{ki}a_{i1}}{a_{kk} - a_{ii}}\right) & X \\
X & \displaystyle\sum_{k=3}^{n} a_{2k}\left(\sum_{\substack{i=1\\ i\neq 2}}^{n} \frac{a_{ki}a_{i2}}{a_{22} - a_{ii}} + \sum_{\substack{i=1\\ i\neq k}}^{n} \frac{a_{ki}a_{i2}}{a_{kk} - a_{ii}}\right) \\
\vdots & \vdots \\
X & X \\
X & X
\end{bmatrix}\right)$$

$$\begin{matrix} \cdots & X & X \\ \cdots & X & X \\ & \vdots & \vdots \\ \cdots & a_{n-1n}\left(\sum\limits_{\substack{i=1\\ i\neq n-1}}^{n}\frac{a_{ni}a_{in-1}}{a_{n-1n-1}-a_{ii}}+\sum\limits_{i=1}^{n-1}\frac{a_{ni}a_{in-1}}{a_{nn}-a_{ii}}\right) & X \\ \cdots & X & 0 \end{matrix}\Bigg]\Bigg),$$

where X are numbers. Simplifying the right hand side of the above expression, we get

$$\begin{aligned} f(0)f'(0) &= \sum_{k=2}^{n} a_{1k}\left(\sum_{i=2}^{n}\frac{a_{ki}a_{i1}}{a_{11}-a_{ii}}+\sum_{\substack{i=1\\ i\neq k}}^{n}\frac{a_{ki}a_{i1}}{a_{kk}-a_{ii}}\right)+\sum_{k=3}^{n} a_{2k}\left(\sum_{\substack{i=1\\ i\neq 2}}^{n}\frac{a_{ki}a_{i2}}{a_{22}-a_{ii}}\right. \\ &\quad \left.+\sum_{\substack{i=1\\ i\neq k}}^{n}\frac{a_{ki}a_{i2}}{a_{kk}-a_{ii}}\right)+\cdots+a_{n-1n}\left(\sum_{\substack{i=1\\ i\neq n-1}}^{n}\frac{a_{ni}a_{in-1}}{a_{n-1n-1}-a_{ii}}+\sum_{i=1}^{n-1}\frac{a_{ni}a_{in-1}}{a_{nn}-a_{ii}}\right) \\ &= a_{12}\left(\frac{a_{22}a_{21}}{a_{11}-a_{22}}+\frac{a_{21}a_{11}}{a_{22}-a_{11}}\right)+\cdots+a_{1n}\left(\frac{a_{nn}a_{n1}}{a_{11}-a_{nn}}+\frac{a_{n1}a_{11}}{a_{nn}-a_{11}}\right) \\ &\quad +a_{23}\left(\frac{a_{33}a_{32}}{a_{22}-a_{33}}+\frac{a_{32}a_{22}}{a_{33}-a_{22}}\right)+\cdots+a_{2n}\left(\frac{a_{nn}a_{n2}}{a_{22}-a_{nn}}+\frac{a_{n2}a_{22}}{a_{nn}-a_{22}}\right) \\ &\quad +\cdots+a_{n-1n}\left(\frac{a_{nn}a_{nn-1}}{a_{n-1n-1}-a_{nn}}+\frac{a_{nn-1}a_{n-1n-1}}{a_{nn}-a_{n-1n-1}}\right)+a_{12}a_{13}a_{23}\left(\frac{1}{a_{11}-a_{33}}\right. \\ &\quad \left.+\frac{1}{a_{22}-a_{33}}+\frac{1}{a_{22}-a_{11}}+\frac{1}{a_{33}-a_{11}}+\frac{1}{a_{11}-a_{22}}+\frac{1}{a_{33}-a_{22}}\right)+\cdots+ \\ &\quad a_{n-2n-1}a_{n-2n}a_{n-1n}\left(\frac{1}{a_{n-2n-2}-a_{nn}}+\frac{1}{a_{n-1n-1}-a_{nn}}+\frac{1}{a_{n-1n-1}-a_{n-2n-2}}\right. \\ &\quad \left.+\frac{1}{a_{nn}-a_{n-2n-2}}+\frac{1}{a_{n-2n-2}-a_{n-1n-1}}+\frac{1}{a_{nn}-a_{n-1n-1}}\right) \\ &= -\sum_{1\leq i<j\leq n} a_{ij}^2 \\ &= -f(0)^2. \end{aligned}$$

So $f'(0) = -f(0)$, provided $f(0) \neq 0$. □

Hence Theorem 3.1.1 implies that Armijo's rule can be applied to the algorithm in (3.4).

A detailed sketch of the algorithm to find the eigenvalues of a symmetric matrix A, whose eigenvalues are all distinct is as follows:

Algorithm 3.1.1 (Symmetric Diagonalization) Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$, whose eigenvalues are all distinct, an orthogonal matrix X_0, and a tolerance *tol* greater than the unit roundoff, the following algorithm computes a diagonalization $X^TAX = \Lambda$, and A is overwritten with the diagonal matrix Λ.

Step 0: Take $X = X_0$. Compute $A^{new} = X^T A^{old} X$, and $f(0) = ||\ \mathrm{lotr}\,(A)\,||_F$.

Step 1: Construct $F = (f_{ij})$, where

$$f_{ij} = \begin{cases} 0 & \text{for } i = j, \\ \dfrac{a_{ij}}{a_{jj} - a_{ii}} & \text{for } i \neq j. \end{cases}$$

Step 2: Let $f(s) = ||G(s)||_F$, where $G(s) = \mathrm{lotr}\left((I + sF)^{-1} A\,(I + sF)\right)$. Evaluate $f(s)$ at $s = 1, \frac{1}{2}, \frac{1}{4}, \cdots$, stopping when

$$f(s) \leq \left(1 - \frac{s}{2}\right) f(0). \tag{3.10}$$

Let t be the first value of s for which (3.10) holds, and let $Y = I + tF$. Let $Y = QR$ be the QR factorization of Y.

Step 3: Update A, and X as follows. $A^{new} = Q^T A^{old} Q$, and $X^{new} = X^{old} Q$.

Step 4: Evaluate $f(0) = ||\mathrm{lotr}\,(A)\,||_F$. Go to Step 1 until $f(0) < tol$, where *tol* is the tolerance for the desired accuracy of the eigenvalues.

Example 3.1.1 If Algorithm 3.1.1 is applied to the matrix

$$A = \begin{bmatrix} 2 & -1 & 3 \\ -1 & 1 & 2 \\ 3 & 2 & -1 \end{bmatrix}$$

with $X_0 = I$, and $tol = 10^{-6}$, then after 6 iterations we obtain a diagonalization $X^TAX = \mathrm{diag}\,(\,2,\ 3.87298,\ -3.87298\,)$. If we use the QR method for a symmetric matrix with single implicit shift to A, then after 5 iterations we obtain a diagonalization $Q^TAQ = \mathrm{diag}(\,-3.87298,\ 3.87298,\ 2\,)$. Next consider the following matrix A_1,

which we obtain by perturbing the elements of A:

$$A_1 = \begin{bmatrix} 2.002 & -1.001 & 3.003 \\ -1.001 & 1.001 & 2.002 \\ 3.003 & 2.002 & -1.004 \end{bmatrix}$$

If we use Algorithm 3.1.1 to A_1 with $X_0 = X$, and $tol = 10^{-6}$, then after 2 iterations we obtain a diagonalization $Y^T A_1 Y = \Lambda$, whereas if we apply the QR method for a symmetric matrix with single implicit shift to $X^T A_1 X$, then after 3 iterations we get a diagonalization $S^T(X^T A_1 X)S = D$.

Example 3.1.2 If Algorithm 3.1.1 is applied to the matrix

$$B = Q\text{diag}(1,2,3,4,5,6)Q^T,$$

where Q is an orthogonal matrix, with $X_0 = I$, and $tol = 10^{-6}$, then after 7 iterations we obtain a diagonalization $X^T BX = \text{diag}(\,5,\,6,\,2,\,1,\,4,\,3\,)$, whereas if we apply the QR method for a symmetric matrix with single implicit shift to B, then after 11 iterations we get a diagonalization $P^T BP = \text{diag}(\,6,\,5,\,1,\,4,\,2,\,3\,)$.

Example 3.1.3 If Algorithm 3.1.1 is applied to the matrix

$$C = \begin{bmatrix} 6 & 4 & 2 & 2 & -1 & 3 \\ 4 & 4 & 1 & 2 & -2 & -3 \\ 2 & 1 & 3 & -2 & -4 & -1 \\ 2 & 2 & -2 & 1 & -6 & 0 \\ -1 & -2 & -4 & -6 & -5 & 4 \\ 3 & -3 & -1 & 0 & 4 & -2 \end{bmatrix}$$

with $X_0 = I$, and $tol = 10^{-6}$, then after 7 iterations we obtain a diagonalization $X^T CX = \Lambda$. If we apply the QR method for a symmetric matrix with single implicit shift to C, then after 10 iterations we get a diagonalization $Q^T CQ = D$. Next consider the following matrix C_1, which we obtain by perturbing the elements of C:

$$C_1 = \begin{bmatrix} 6.004 & 4.006 & 2.007 & 2.006 & -1.002 & 3.007 \\ 4.006 & 4.007 & 1.004 & 2.009 & -2.003 & -3.004 \\ 2.007 & 1.004 & 3.001 & -2.002 & -4.004 & -1.001 \\ 2.006 & 2.009 & -2.002 & 1.002 & -6.008 & 0.004 \\ -1.002 & -2.003 & -4.004 & -6.008 & -5.007 & 4.001 \\ 3.007 & -3.004 & -1.001 & 0.004 & 4.001 & -2.004 \end{bmatrix}$$

If we use Algorithm 3.1.1 to C_1 with $X_0 = X$, and $tol = 10^{-6}$, then after 3 iterations we obtain a diagonalization $Y^T A_1 Y = \Omega$, whereas if we apply the QR method for a symmetric matrix with single implicit shift to $X^T C_1 X$, then after 10 iterations we obtain a diagonalization $S^T (X^T C_1 X) S = \Gamma$.

3.2 Block Diagonalization of a Symmetric Matrix using Armijo's Stepsize

Let A be an $n \times n$ symmetric matrix. Our goal is to find an iterative method to compute a block diagonalization $A = Q \Lambda Q^T$, where $\Lambda = \text{diag}(\Lambda_1, \Lambda_2, \ldots, \Lambda_k)$ is a block diagonal matrix, and $Q = [Q_1, Q_2, \ldots, Q_k]$ is a compatible block column orthogonal matrix such that $AQ_j = Q_j \Lambda_j$. Here $\Lambda_j = \text{diag}(\lambda_j, \lambda_j, \ldots, \lambda_j)$ is an $m_j \times m_j$ scalar matrix. It can always be arranged so that the eigenvalues of Λ_i, and Λ_j for $i \neq j$ are distinct.

From the block diagonalization of a matrix A in Section 2.1, we have the following locally quadratically convergent algorithm:

$$\begin{aligned} \Lambda_j^{new} &= \left(Y_j^{old}\right)^T A X_j^{old} \qquad \text{for } j = 1 \text{ to } k, \\ X_j^{new} &= X_j^{old} + \sum_{\substack{i=1 \\ i \neq j}}^{k} X_i^{old} P_{ij} \quad \text{for } j = 1 \text{ to } k, \\ Y^{new} &= \left(X^{new}\right)^{-1}, \end{aligned} \tag{3.11}$$

where $P = P_{ij}$ is the solution to the matrix equation $P \Lambda_j^{new} - \Lambda_i^{new} P = \left(Y_i^{old}\right)^T A X_j^{old}$, and $\left(Y_j^{old}\right)^T$ is the jth block row of Y^{old}. Since Λ_i^{new}, and Λ_j^{new} are scalar matrices, so $P_{ij} = \dfrac{\left(Y_i^{old}\right)^T A X_j^{old}}{\lambda_j^{new} - \lambda_i^{new}}$. After simplifying the algorithm in (3.11) to compute a block diagonalization of a symmetric matrix A, we have

$$\begin{aligned} \Lambda^{new} &= \text{Diag}\left(\left(X^{old}\right)^{-1} A X^{old}\right), \\ X^{new} &= X^{old}\left(I + F\left(X^{old}, \Lambda^{new}\right)\right), \end{aligned} \tag{3.12}$$

where $\text{Diag}(B) = \text{diag}(B_{11}, B_{22}, \ldots, B_{kk})$ with $B_{jj} = \text{diag}(b_j, b_j, \ldots, b_j)$ is an $m_j \times m_j$ scalar matrix, and $F\left(X^{old}, \Lambda^{new}\right)$ is a block matrix such that $F_{jj}\left(X^{old}, \Lambda^{new}\right)$

is an $m_j \times m_j$ zero matrix, and

$$F_{ij}\left(X^{old}, \Lambda^{new}\right) = \frac{\left(\left(X^{old}\right)^{-1} A X^{old}\right)_{ij}}{\lambda_j^{new} - \lambda_i^{new}} \quad \text{for } i \neq j,$$

where $\left(\left(X^{old}\right)^{-1} A X^{old}\right)_{ij}$ is the (i,j) block of $\left(X^{old}\right)^{-1} A X^{old}$. In (3.12), if we put $Y^{new} = I + F\left(X^{old}, \Lambda^{new}\right)$, and $A^{old} = \left(X^{old}\right)^{-1} A X^{old}$, then the algorithm reduces to

$$\begin{aligned} \Lambda^{new} &= \text{Diag}\left(\left(Y^{old}\right)^{-1} A^{old} Y^{old}\right), \\ X^{new} &= X^{old} Y^{new}. \end{aligned}$$

Hence the iterative method to find the eigenvalues, and corresponding eigenvectors of a matrix A can be rewritten as follows:

$$\begin{aligned} A^{new} &= \left(Y^{old}\right)^{-1} A^{old} Y^{old}, \\ \Lambda^{new} &= \text{Diag}\left(A^{new}\right), \\ Y^{new} &= I + F\left(A^{new}\right), \\ X^{new} &= X^{old} Y^{new}, \end{aligned} \tag{3.13}$$

where A^{new} is a $k \times k$ block matrix with $A_{jj}^{new} = \text{diag}\left(a_j^{new}, a_j^{new}, \ldots, a_j^{new}\right)$, $F_{jj}\left(A^{new}\right)$ is a $m_j \times m_j$ zero matrix, and $F_{ij}\left(A^{new}\right) = \dfrac{A_{ij}^{new}}{a_j^{new} - a_i^{new}}$ for $i \neq j$.

If the starting guess matrix of eigenvectors X_0 is not a good approximation of Q in the factorization $A = Q\Lambda Q^T$, then the increment $F\left(A^{new}\right)$ in $Y^{new} = I + F\left(A^{new}\right)$ may be too large. However with a large increment, we may likely have instabilitiy in the updating process, and the algorithm may diverge. To restore the convergence of the algorithm, and to have a steady change in the value of Y^{new}, we need a small increment in each iteration. To achieve this, we redefine each iterate, and introduce a small positive parameter. Let s be a positive parameter, and define $Z(s)^{new} = I + sF\left(A^{new}\right)$. Then $Z(0)^{new} = I$, and $Z(1)^{new} = Y^{new}$, the matrix generated by (3.13). Define $G(s) = \text{nondiag}\left(\left(Z(s)^{old}\right)^{-1} A^{old} Z(s)^{old}\right)$, and $f(s) = ||G(s)||_F$, where nondiag(B) is a block matrix formed from the $k \times k$ block matrix B by replacing the diagonal blocks by zero matrices. Suppressing the superscripts in

$Z(s)^{new}$, and $G(s)$, we have $Z(s) = I + sF(A)$, and $G(s) = \text{nondiag}\,(Z(s)^{-1}AZ(s))$. Hence $f(0) = ||\text{nondiag}\,(A)\,||_F$, and $f(1) = ||\text{nondiag}\,(Y^{-1}AY)\,||_F$.

With a starting guess X_0, if the iterations in (3.13) converge, then we must have $f(1) \leq f(0)$. As in the previous section, our goal is to find an s, $0 < s \leq 1$, for which $f(s) \leq f(0)$ holds. To this end, we use Armijo's rule from optimization theory. In Armijo's rule, we determine s in the following way. Evaluate $f(s)$ at $s = 1, \frac{1}{2}, \frac{1}{4}, \ldots$, stopping when

$$f(s) \leq \left(1 - \frac{s}{2}\right) f(0).$$

To use the above inequality we must make sure that $f(s)$ satisfies $f'(0) = -f(0)$. Hence our next aim is to determine $f'(0)$, when $f(s) = ||G(s)||_F$, and $G(s) = \text{nondiag}\left((Z(s))^{-1}\,AZ(s)\right)$.

Theorem 3.2.1 *Let $A \in \mathbb{R}^{n\times n}$ be a $k \times k$ block symmetric matrix such that $A_{jj} = diag(\,a_j, a_j, \ldots, a_j\,)$ is an $m_j \times m_j$ scalar matrix. If we define $Z(s) = I + sF(A)$, $G(s) = nondiag\left((Z(s))^{-1}\,AZ(s)\right)$, and $f(s) = ||G(s)||_F$, where $F_{jj}(A)$ is an $m_j \times m_j$ zero matrix, $F_{ij}(A) = \dfrac{A_{ij}}{a_j - a_i}$ for $i \neq j$, and $nondiag(B)$ is formed from the $k \times k$ block matrix B by replacing the diagonal blocks by zero matrices, then $f'(0) = -f(0)$.*

Proof of Theorem 3.2.1 The result (2.18) of Lemma 2.3.1 gives

$$f(0)\frac{d}{ds}f(s)|_{s=0} = \text{trace}\left(G(0)^T\frac{d}{ds}G(s)|_{s=0}\right). \tag{3.14}$$

Differentiating $G(s)$, and $Z(s)$ with respect to s, and then evaluating the derivatives at $s = 0$, we get

$$\begin{aligned}\frac{d}{ds}G(s)|_{s=0} &= \text{nondiag}\left(-Z(0)^{-1}\frac{d}{ds}Z(s)|_{s=0}Z(0)^{-1}AZ(0) + Z(0)^{-1}A\frac{d}{ds}Z(s)|_{s=0}\right),\\ \frac{d}{ds}Z(s)|_{s=0} &= F.\end{aligned} \tag{3.15}$$

Since $Z(0) = I$, so after simplification (3.15) gives

$$\frac{d}{ds}G(s)|_{s=0} = \text{nondiag}\,(AF - FA)\,. \tag{3.16}$$

Next $G(0) = \text{nondiag}\,(Z(0)^{-1}AZ(0)) = \text{nondiag}\,(IAI) = \text{nondiag}(A)$. Hence from (3.14), we obtain

$$f(0)\frac{d}{ds}f(s)|_{s=0} = \text{trace}\left((\text{nondiag}(A))^T\,\text{nondiag}\,(AF - FA)\right). \tag{3.17}$$

But A is symmetric, so $(\text{nondiag}(A))^T = \text{nondiag}(A)$. Hence (3.17) reduces to

$$f(0)\frac{d}{ds}f(s)|_{s=0} = \text{trace}\,(\text{nondiag}(A)\text{nondiag}(AF - FA))\,. \tag{3.18}$$

Next, for $i \neq j$, $F_{ij} = \dfrac{A_{ij}}{a_j - a_i}$, and $F_{ji} = \dfrac{A_{ji}}{a_i - a_j}$. Since A is symmetric, so $F_{ij} = -F_{ji}^T$. That is F is a skew symmetric matrix, and therefore by Lemma 3.1.1, $AF - FA$ is a symmetric matrix. Next

$$AF - FA =$$

$$\begin{bmatrix}
\displaystyle\sum_{i=2}^{k}\frac{2\,A_{1i}A_{i1}}{a_1 - a_i} & \displaystyle\sum_{\substack{i=1\\ i\neq 2}}^{k}\frac{A_{1i}A_{i2}}{a_2 - a_i} + \sum_{i=2}^{k}\frac{A_{1i}A_{i2}}{a_1 - a_i} & \cdots & \displaystyle\sum_{i=1}^{k-1}\frac{A_{1i}A_{ik}}{a_k - a_i} + \sum_{i=2}^{k}\frac{A_{1i}A_{ik}}{a_1 - a_i} \\
\displaystyle\sum_{i=2}^{k}\frac{A_{2i}A_{i1}}{a_1 - a_i} + \sum_{\substack{i=1\\ i\neq 2}}^{k}\frac{A_{2i}A_{i1}}{a_2 - a_i} & \displaystyle\sum_{\substack{i=1\\ i\neq 2}}^{k}\frac{2\,A_{2i}A_{i2}}{a_2 - a_i} & \cdots & \displaystyle\sum_{i=1}^{k-1}\frac{A_{2i}A_{ik}}{a_k - a_i} + \sum_{\substack{i=1\\ i\neq 2}}^{k}\frac{A_{2i}A_{ik}}{a_2 - a_i} \\
\vdots & \vdots & & \vdots \\
\displaystyle\sum_{i=2}^{k}\frac{A_{ki}A_{i1}}{a_1 - a_i} + \sum_{i=1}^{k-1}\frac{A_{ki}A_{i1}}{a_k - a_i} & \displaystyle\sum_{\substack{i=1\\ i\neq 2}}^{k}\frac{A_{ki}A_{i2}}{a_2 - a_i} + \sum_{i=1}^{k-1}\frac{A_{ki}A_{i2}}{a_k - a_i} & \cdots & \displaystyle\sum_{i=1}^{k-1}\frac{2\,A_{ki}A_{ik}}{a_k - a_i}
\end{bmatrix}.$$

Therefore from (3.18), we have

$$f(0)\frac{d}{ds}f(s)|_{s=0} =$$

$$\text{trace}\left(\left[\begin{array}{ccc}
\displaystyle\sum_{j=2}^{k} A_{1j}\left(\sum_{i=2}^{k}\frac{A_{ji}A_{i1}}{a_1-a_i}+\sum_{\substack{i=1\\ i\neq j}}^{k}\frac{A_{ji}A_{i1}}{a_j-a_i}\right) & & X \\
& X & \displaystyle\sum_{\substack{j=1\\ j\neq 2}}^{k} A_{2j}\left(\sum_{\substack{i=1\\ i\neq 2}}^{k}\frac{A_{ji}A_{i2}}{a_2-a_i}+\sum_{\substack{i=1\\ i\neq j}}^{k}\frac{A_{ji}A_{i2}}{a_j-a_i}\right) \\
& \vdots & \vdots \\
& X & X \\
& X & X \\
\cdots & X & X \\
\cdots & X & X \\
& \vdots & \vdots \\
\cdots \displaystyle\sum_{\substack{j=1\\ j\neq k-1}}^{k} A_{k-1j}\left(\sum_{\substack{i=1\\ i\neq k-1}}^{k}\frac{A_{ji}A_{ik-1}}{a_{k-1}-a_i}+\sum_{\substack{i=1\\ i\neq j}}^{k}\frac{A_{ji}A_{ik-1}}{a_j-a_i}\right) & & X \\
\cdots & X & \displaystyle\sum_{j=1}^{k-1} A_{kj}\left(\sum_{i=1}^{k-1}\frac{A_{ji}A_{ik}}{a_k-a_i}+\sum_{\substack{i=1\\ i\neq j}}^{k}\frac{A_{ji}A_{ik}}{a_j-a_i}\right)
\end{array}\right]\right),$$

where X are rectangular matrices. After simplifying the right hand side of the above expression, we obtain

$$\begin{aligned}
f(0)\frac{d}{ds}f(s)|_{s=0} &= 2\left[\text{trace}\left(\sum_{j=2}^{k} A_{1j}\left(\sum_{i=2}^{k}\frac{A_{ji}A_{i1}}{a_1-a_i}+\sum_{\substack{i=1\\ i\neq j}}^{k}\frac{A_{ji}A_{i1}}{a_j-a_i}\right)\right)\right. \\
&\quad +\text{trace}\left(\sum_{j=3}^{k} A_{2j}\left(\sum_{\substack{i=1\\ i\neq 2}}^{k}\frac{A_{ji}A_{i2}}{a_2-a_i}+\sum_{\substack{i=1\\ i\neq j}}^{k}\frac{A_{ji}A_{i2}}{a_j-a_i}\right)\right) \\
&\quad \left.+\text{trace}\left(A_{k-1k}\left(\sum_{\substack{i=1\\ i\neq k-1}}^{k}\frac{A_{ki}A_{ik-1}}{a_{k-1}-a_i}+\sum_{i=1}^{k-1}\frac{A_{ki}A_{ik-1}}{a_k-a_i}\right)\right)\right] \\
&= 2\left[\text{trace}\left(A_{12}\left(\frac{A_{22}A_{21}}{a_1-a_2}+\frac{A_{21}A_{11}}{a_2-a_1}\right)\right)+\cdots\right. \\
&\quad +\text{trace}\left(A_{1k}\left(\frac{A_{kk}A_{k1}}{a_1-a_k}+\frac{A_{k1}A_{11}}{a_k-a_1}\right)\right) \\
&\quad +\text{trace}\left(A_{23}\left(\frac{A_{33}A_{32}}{a_2-a_3}+\frac{A_{32}A_{22}}{a_3-a_2}\right)\right)+\cdots \\
&\quad \left.+\text{trace}\left(A_{2k}\left(\frac{A_{kk}A_{k2}}{a_2-a_k}+\frac{A_{k2}A_{22}}{a_k-a_2}\right)\right)+\cdots+\right.
\end{aligned}$$

$$\begin{aligned}
&\operatorname{trace}\left(A_{k-1k}\left(\frac{A_{kk}A_{kk-1}}{a_{k-1}-a_k}+\frac{A_{kk-1}A_{k-1k-1}}{a_k-a_{k-1}}\right)\right)+\cdots\\
&+\operatorname{trace}\left(A_{12}\left(\frac{A_{23}A_{31}}{a_1-a_3}+\frac{A_{23}A_{31}}{a_2-a_3}\right)\right)+\operatorname{trace}\left(A_{13}\left(\frac{A_{32}A_{21}}{a_1-a_2}+\frac{A_{32}A_{21}}{a_3-a_2}\right)\right)\\
&+\operatorname{trace}\left(A_{23}\left(\frac{A_{31}A_{12}}{a_2-a_1}+\frac{A_{31}A_{12}}{a_3-a_1}\right)\right)+\cdots\\
&+\operatorname{trace}\left(A_{k-2k-1}\left(\frac{A_{k-1k}A_{kk-2}}{a_{k-2}-a_k}+\frac{A_{k-1k}A_{kk-1}}{a_{k-1}-a_k}\right)\right)\\
&+\operatorname{trace}\left(A_{k-2k}\left(\frac{A_{kk-1}A_{k-1k-2}}{a_{k-2}-a_{k-1}}+\frac{A_{kk-1}A_{k-1k-2}}{a_k-a_{k-1}}\right)\right)\\
&+\operatorname{trace}\left(A_{k-1k}\left(\frac{A_{kk-2}A_{k-2k-1}}{a_{k-1}-a_{k-2}}+\frac{A_{kk-2}A_{k-2k-1}}{a_k-a_{k-2}}\right)\right)\Bigg]. \qquad (3.19)
\end{aligned}$$

Since $A_{ij}=A_{ji}^T$, A_{jj}, and A_{ii} are scalar matrices, so $A_{jj}A_{ji}=a_jA_{ji}$, $A_{ji}A_{ii}=a_iA_{ji}$, and

$$A_{ij}\left(\frac{A_{jj}A_{ji}}{a_i-a_j}+\frac{A_{ji}A_{ii}}{a_j-a_i}\right)=A_{ij}\left(\frac{a_jA_{ji}}{a_i-a_j}-\frac{a_iA_{ji}}{a_i-a_j}\right)=-A_{ji}^TA_{ji}.$$

Hence

$$\operatorname{trace}\left(A_{ij}\left(\frac{A_{jj}A_{ji}}{a_i-a_j}+\frac{A_{ji}A_{ii}}{a_j-a_i}\right)\right) \;=\; -\operatorname{trace}\left(A_{ji}^TA_{ji}\right). \qquad (3.20)$$

Next we know that for any matrices C, and D, $\operatorname{trace}(C)=\operatorname{trace}\left(C^T\right)$, and $\operatorname{trace}(CD)=\operatorname{trace}(DC)$, provided the matrix multiplications exist. Let $\Theta=\operatorname{trace}(A_{ij}A_{jl}A_{li})$. Then clearly $\Theta=\operatorname{trace}\left((A_{ij}A_{jl}A_{li})^T\right)=\operatorname{trace}\left(A_{li}^TA_{jl}^TA_{ij}^T\right)=\operatorname{trace}(A_{il}A_{lj}A_{ji})$. Also $\Theta=\operatorname{trace}(A_{ij}(A_{jl}A_{li}))=\operatorname{trace}((A_{jl}A_{li})A_{ij})$. Hence

$$\begin{aligned}
&\operatorname{trace}\left(A_{ij}\left(\frac{A_{jl}A_{li}}{a_i-a_l}+\frac{A_{jl}A_{li}}{a_j-a_l}\right)\right)+\operatorname{trace}\left(A_{il}\left(\frac{A_{lj}A_{ji}}{a_i-a_j}+\frac{A_{lj}A_{ji}}{a_l-a_j}\right)\right)\\
&+\operatorname{trace}\left(A_{jl}\left(\frac{A_{li}A_{ij}}{a_j-a_i}+\frac{A_{li}A_{ij}}{a_l-a_i}\right)\right)\\
=\;&\Theta\left(\frac{1}{a_i-a_l}+\frac{1}{a_j-a_l}+\frac{1}{a_i-a_j}+\frac{1}{a_l-a_j}+\frac{1}{a_j-a_i}+\frac{1}{a_l-a_i}\right)\\
=\;&\Theta\cdot 0=0. \qquad (3.21)
\end{aligned}$$

Using the results from (3.20), and (3.21) in (3.19), we have

$$f(0)\frac{d}{ds}f(s)|_{s=0} \;=\; -2\left[\operatorname{trace}\left(A_{21}^TA_{21}\right)+\cdots+\operatorname{trace}\left(A_{k1}^TA_{k1}\right)+\operatorname{trace}\left(A_{32}^TA_{32}\right)\right.$$

$$+ \cdots + \text{trace}\left(A_{k2}^T A_{k2}\right) + \cdots + \text{trace}\left(A_{kk-1}^T A_{kk-1}\right)\Big] . \qquad (3.22)$$

But

$$\begin{aligned} f(0)^2 &= \text{trace}\left(G(0)^T G(0)\right) \\ &= \text{trace}\left(\text{nondiag}(A)\text{nondiag}(A)\right) \\ &= 2\Big[\text{trace}\left(A_{21}^T A_{21}\right) + \cdots + \text{trace}\left(A_{k1}^T A_{k1}\right) + \text{trace}\left(A_{32}^T A_{32}\right) \\ &\quad + \cdots + \text{trace}\left(A_{k2}^T A_{k2}\right) + \cdots + \text{trace}\left(A_{kk-1}^T A_{kk-1}\right)\Big] . \end{aligned}$$

Hence (3.22) yields $f(0)\frac{d}{ds}f(s)|_{s=0} = -f(0)^2$. So $f'(0) = -f(0)$, provided $f(0) \neq 0$. □

The result of the above theorem implies that Armijo's rule can be applied to the algorithm in (3.13).

In Theorem 3.2.1, $f'(0) = -f(0)$ holds only when A is symmetric. So we need to restore the symmetry of A in the update $A^{new} = \left(Y^{old}\right)^{-1} A^{old} Y^{old}$ in (3.13). If Y^{old} is an orthogonal matrix, then A^{new} will be symmetric. In general, Y^{old} is not an orthogonal matrix. Let $Y^{new} = Q^{new} R^{new}$ be a QR factorization of Y^{new}. Now in (3.13), if we replace $A^{new} = \left(Y^{old}\right)^{-1} A^{old} Y^{old}$ by $A^{new} = \left(Q^{old}\right)^T A^{old} Q^{old}$, and $X^{new} = X^{old} Y^{new}$ by $X^{new} = X^{old} Q^{new}$, then A^{new} will retain the symmetry, and X^{new} will retain the orthogonality.

A sketch of an algorithm to find the eigenvalues of a symmetric matrix A is as follows:

Algorithm 3.2.1 (Symmetric Block Diagonalization) Given an $n \times n$ symmetric matrix A, an orthogonal matrix X_0, a tolerance *tol* greater than the unit roundoff, and a coalescing tolerance *tol1* greater than the square root of the unit roundoff, the following algorithm overwrites A with $\Lambda = X^T A X$, where Λ is a block diagonal matrix, and X is an orthogonal matrix.

Step 0: Take $X = X_0$. Compute $A^{new} = X^T A^{old} X$, and $f(0) = ||\ \text{nondiag}\,(A)\,||_F$, where nondiag (A) is defined as earlier.

Step 1: Let $NB = n$ (the number of diagonal blocks of A), $NS = [\,1, 1, \ldots, 1\,]$; an array of n elements whose ith element is the dimension of the ith diagonal block of A. For $i < j$, let a_i be the eigenvalue of A_{ii}, and a_j be the eigenvalue of A_{jj}. If $|a_i - a_j| < tol1$, then merge blocks A_{ii} and A_{jj} to form a single block using permutations of columns and rows of A. Use the same column permutations to X to merge block columns X_i and X_j. Update NB^{old}, and NS^{old} to get NB^{new}, and NS^{new}. Try the above merging procedure for all possible combinations of $1 \leq i < j \leq n$. Using column and row permutations arrange the blocks of A such that the diagonal blocks appear in the decreasing order of sizes along the diagonal. Use those column permutations to arrange block columns of X in the decreasing order of sizes as well. Take $\Lambda = \text{diag}(\, A_{11}, A_{22}, \ldots, A_{NBNB}\,)$.

Step 2: Construct block matrix $F = (F_{ij})$ as follows:

Take $F_{jj} = 0$, which is an $m_j \times m_j$ zero matrix, and for $i \neq j$, F_{ij} can be determined by using the following loop.

```
for j = 1 : NB
    for i = 1 : NB
        if i ≠ j
            Solve Z A_jj − A_ii Z = A_ij for Z.
            F_ij = Z
        end
    end
end
```

Step 3: Let $f(s) = ||G(s)||_F$, where $G(s)$ =nondiag$\Big((I+sF)^{-1}A(I+sF)\Big)$. Evaluate $f(s)$ at $s = 1, \frac{1}{2}, \frac{1}{4}, \ldots$, stopping when

$$f(s) \leq \left(1 - \frac{s}{2}\right) f(0). \tag{3.23}$$

Let t be the first value of s for which (3.23) holds. Let $Z = I + tF$, and let $Z = QR$ be a QR factorization of Z.

Step 4 : Update A, and X as follows: $A^{new} = Q^T A^{old} Q$, and $X^{new} = X^{old} Q$, where Q is the orthogonal matrix from Step 3.

Step 5: Compute $f(0) = ||\text{nondiag}\,(A)\,||_F$, and goto Step 1 until $f(0) < tol$.

Example 3.2.1 If Algorithm 3.2.1 is applied to the matrix

$$A = \begin{bmatrix} 13 & -4 & 2 \\ -4 & 13 & -2 \\ 2 & -2 & 10 \end{bmatrix}$$

with $X_0 = I$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 5 iterations we obtain a diagonalization $X^T A X = \text{diag}(\ 9,\ 9,\ 18\ )$. If we use the QR method for a symmetric matrix with single implicit shift to A, then after 2 iterations we get a diagonalization $S^T A S = \text{diag}(\ 9,\ 18,\ 9\ )$. Next consider the following matrix A_1, which we obtain by perturbing the elements of A:

$$A_1 = \begin{bmatrix} 13.04 & -4.03 & 2.02 \\ -4.03 & 13.02 & -2.01 \\ 2.02 & -2.01 & 10.03 \end{bmatrix}$$

If we use Algorithm 3.2.1 to A_1 with $X_0 = X$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 4 iterations we obtain a diagonalization $Y^T A_1 Y = \Lambda$. If we apply the QR method for a symmetric matrix with single implicit shift to $X^T A_1 X$, then also after 4 iterations we obtain a diagonalization $Q^T(X^T A_1 X)Q = D$.

Example 3.2.2 If Algorithm 3.2.1 is applied to the matrix

$$B = Q\text{diag}(2,2,2,3,3,3)Q^T,$$

where Q is an orthogonal matrix with $X_0 = I$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 10 iterations we obtain a diagonalization $X^TBX = \text{diag}(3,3,3,2,2,2)$, whereas if we use the QR method for a symmetric matrix with single implicit shift to B, then after 4 iterations we get a diagonalization $S^TBS = \text{diag}(\ 3,\ 2,\ 3,\ 2,\ 2,\ 3\ )$.

Example 3.2.3 If Algorithm 3.2.1 is applied to Rosser matrix

$$R = \begin{bmatrix} 611 & 196 & -192 & 407 & -8 & -52 & -49 & 29 \\ 196 & 899 & 113 & -192 & -71 & -43 & -8 & -44 \\ -192 & 113 & 899 & 196 & 61 & 49 & 8 & 52 \\ 407 & -192 & 196 & 611 & 8 & 44 & 59 & -23 \\ -8 & -71 & 61 & 8 & 411 & -599 & 208 & 208 \\ -52 & -43 & 49 & 44 & -599 & 411 & 208 & 208 \\ -49 & -8 & 8 & 59 & 208 & 208 & 99 & -911 \\ 29 & -44 & 52 & -23 & 208 & 208 & -911 & 99 \end{bmatrix}$$

with $X_0 = I$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 10 iterations we obtain a diagonalization $X^TRX = \text{diag}(\ 1000,\ 1000,\ 1020,\ 0.09805, -1020.04902,\ 0.\ 1020.04902,\ 1019.90195\ )$. If we apply the QR method for a symmetric matrix with single implicit shift to R, then after 7 iterations we get a diagonalization $S^TRS = \text{diag}(\ -1020.04902,\ 0.09805,\ 1000,\ 0,\ 1020.04902,\ 1020,\ 1019.90195,\ 1000\ )$. Next consider the matrix

$$P = \begin{bmatrix} 96 & 15 & -65 & 14 & -98 & -75 & -77 & 11 \\ 15 & 53 & 38 & -50 & -80 & -37 & -39 & -75 \\ -65 & 38 & 12 & 99 & 96 & 20 & 62 & 40 \\ 14 & -50 & 99 & 89 & 1 & 36 & 73 & -81 \\ -98 & -80 & 96 & 1 & 10 & -39 & 28 & 29 \\ -75 & -37 & 20 & 36 & -39 & 47 & 31 & 28 \\ -77 & -39 & 62 & 73 & 28 & 31 & 69 & -59 \\ 11 & -75 & 40 & -81 & 29 & 28 & -59 & 30 \end{bmatrix}$$

If we apply Algorithm 3.2.1 to the perturbed matrix $R_1 = R + P/1000$ with $X_0 = X$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 5 iterations we obtain a diagonalization

$Y^T R_1 Y = \Lambda$, whereas if we apply the QR method for a symmetric matrix with single implicit shift to $X^T R_1 X$, then after 8 iterations we get a diagonalization $Q^T(X^T R_1 X)Q = D$.

Example 3.2.4 If Algorithm 3.2.1 is applied to Wilkinson's 21×21 test matrix

$$W = \begin{bmatrix}
10 & 1 & & & & & & & & & & & & & & & & & & & \\
1 & 9 & 1 & & & & & & & & & & & & & & & & & & \\
 & 1 & 8 & 1 & & & & & & & & & & & & & & & & & \\
 & & 1 & 7 & 1 & & & & & & & & & & & & & & & & \\
 & & & 1 & 6 & 1 & & & & & & & & & & & & & & & \\
 & & & & 1 & 5 & 1 & & & & & & & & & & & & & & \\
 & & & & & 1 & 4 & 1 & & & & & & & & & & & & & \\
 & & & & & & 1 & 3 & 1 & & & & & & & & & & & & \\
 & & & & & & & 1 & 2 & 1 & & & & & & & & & & & \\
 & & & & & & & & 1 & 1 & 1 & & & & & & & & & & \\
 & & & & & & & & & 1 & 0 & 1 & & & & & & & & & \\
 & & & & & & & & & & 1 & 1 & 1 & & & & & & & & \\
 & & & & & & & & & & & 1 & 2 & 1 & & & & & & & \\
 & & & & & & & & & & & & 1 & 3 & 1 & & & & & & \\
 & & & & & & & & & & & & & 1 & 4 & 1 & & & & & \\
 & & & & & & & & & & & & & & 1 & 5 & 1 & & & & \\
 & & & & & & & & & & & & & & & 1 & 6 & 1 & & & \\
 & & & & & & & & & & & & & & & & 1 & 7 & 1 & & \\
 & & & & & & & & & & & & & & & & & 1 & 8 & 1 & \\
 & & & & & & & & & & & & & & & & & & 1 & 9 & 1 \\
 & & & & & & & & & & & & & & & & & & & 1 & 10
\end{bmatrix}$$

with $X_0 = I$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 10 iterations we obtain a diagonalization $X^T W X = \text{diag}($ 10.74619, 10.74619, 9.21068, 9.21068, 8.03894, 8.03894, 7.00395, 7.00395, 6.00023, 6.00023, 5.00001, 5.00001, 3.99605, 4.00435, 3.04310, 2.96106, 2.13021, 1.78932, 0.94753, 0.25381, -1.12544), whereas if we use the QR method for a symmetric matrix with single implicit shift to W, then after 32 iterations we obtain a diagonalization $S^T W S = \text{diag}($ -1.12544, 0.25381, 0.94753, 1.78932, 2.13021, 2.96106, 3.04310, 3.99605, 4.00435, 7.00395, 5.00024, 4.99978, 6.00023, 6.00023, 7.00395, 8.03894, 8.03894, 9.21068, 9.21068, 10.74619, 10.74619).

Example 3.2.5 If Algorithm 3.2.1 is applied to Wilkinson's 7×7 test matrix

$$W = \begin{bmatrix} 3 & 1 & & & & & \\ 1 & 2 & 1 & & & & \\ & 1 & 1 & 1 & & & \\ & & 1 & 0 & 1 & & \\ & & & 1 & 1 & 1 & \\ & & & & 1 & 2 & 1 \\ & & & & & 1 & 3 \end{bmatrix}$$

with $X_0 = I$, $tol = 10^{-6}$, and $tol1 = 10^{-4}$, then after 9 iterations we obtain a diagonalization $X^T W X = \text{diag}(3.76156, 3.73205, 2.36333, 2, 1, 0.26795, -1.12489)$, whereas if we use the QR method for a symmetric matrix with single implicit shift to W, then after 12 iterations we get a diagonalization $S^T W S = \text{diag}(-1.12489, 0.26795, 1, 2, 2.36333, 0.76156, 3.73205)$.

Example 3.2.6 If Algorithm 3.2.1 is applied to Wilkinson's 21×21 test matrix

Note that, if at least two diagonal elements of the given matrix A are equal, and the off diagonal elements are large compared to the diagonal elements, then Algorithm 3.2.1 can not be applied directly to A. In this case, first we find a Hessenberg form H of A, and then apply Algorithm 3.2.1 to the Hessenberg form H.

If we apply Algorithm 3.2.1 to an $n \times n$ symmetric matrix A with $X_0 = I$, a tolerance tol, and a coalescing tolerance $tol1$, then we obtain a diagonalization $X^T A X = \Lambda$. Next we can use X as the starting guess orthogonal matrix to obtain a diagonalization of a slightly perturbed symmetric matrix $A + \epsilon E$ with very rapid convergence, where E is an arbitrary symmetric matrix, and ϵ is a small scalar. In the following table for different values of n we give the number of iterations required to obtain diagonalizations of $A + \epsilon E$ by Algorithm 3.2.1 with $X_0 = X$, and diagonalizations of $X^T(A + \epsilon E)X$ by the QR method for a symmetric matrix with single implicit shift. Here the original matrices $A = (a_{ij})$ are random symmetric matrices,

where a_{ij} are integers and $0 \leq a_{ij} \leq 20$, and the matrices $E = (e_{ij})$ are random symmetric matrices, where e_{ij} are real numbers and $0 < e_{ij} < 2$. The tolerance is $tol = 10^{-6}$, and the coalescing tolerance is $tol1 = 10^{-4}$:

size	number of iterations for our algorithm	number of iterations for the QR algorithm with single shift		ϵ
	$A + \epsilon E$	$A + \epsilon E$	$S^T(A + \epsilon E)S$	
10	3	16	15	10^{-2}
20	3	33	32	10^{-2}
30	3	50	50	10^{-2}
40	3	64	64	10^{-2}
50	3	79	80	10^{-2}

CHAPTER 4
CONCLUSION

In the introduction, we reviewed the application of eigenvalue sensitivity, and eigenvector sensitivity to derive an iterative method to find all eigenvalues, and corresponding eigenvectors of a matrix A, whose eigenvalues are all distinct. To use this method, we need a matrix of approximate eigenvectors X_0. If X_0 is not a good approximation of X in the factorization $A = X\Lambda X^{-1}$, where Λ is a diagonal matrix, then the method diverges. As discussed in Section 2.1, the application of eigenvaluesensitivity, and eigenvector sensitivity can be theoretically extended to derive an iterative method to find all eigenvalues, and corresponding eigenvectors of a nondefective matrix A, some of whose eigenvalues may be identical. The obvious difficulty one encounters in using this method is how to choose a matrix of approximate eigenvectors for the starting guess. In Section 2.4, we showed that the block diagonalization method converges locally quadratically. If by another method, a matrix of eigenvectors X of a matrix A can be determined (in Section 2.1, we applied the QR method with double implicit shift to obtain the eigenvalues, and then solve $Bx = 0$ for x, where $B = A - \lambda_i I$ to obtain a corresponding eigenvector to the eigenvalue λ_i, $i = 1, \ldots, n$. To solve $Bx = 0$ for x, we assume $x(i) = 1$, move the ith column of B to the right hand side of the equal sign, and then solve an overdetermined system of equations by the QR factorization technique,) then the block diagonalization method can be used to find the eigenvalues of a slightly perturbed matrix $A + \epsilon E$ with X as the starting guess matrix of eigenvectors with rapid convergence, where E is an arbitrary matrix, and ϵ is a small scalar.

In the Differential Equation Appoach to Eigencomputations, we use the Euler method. The differential equations that we solve for the eigenproblem are:

$$\dot{\Lambda}(t) = -\Lambda(t) + \text{Diag}\left(X(t)^{-1}AX(t)\right), \text{ and } \dot{X}(t) = X(t)F\left(X(t), \Lambda(t)\right),$$

where Diag(M) is a block diagonal matrix formed from the diagonal blocks of the block matrix M, $F_{jj}\left(X(t), \Lambda(t)\right)$ is a square zero matrix, and $F_{ij}\left(X(t), \Lambda(t)\right)$ for $i \neq j$ is the solution B to the matrix equation $B\Lambda_j(t) - \Lambda_i(t)B = Y_i(t)^T AX_j(t)$. Here $Y_i(t)^T$ denotes the ith block row of $X(t)^{-1}$. The iterates to update Λ, and X are:

$$\Lambda_{n+1}(t) = \Lambda_n + \Delta t\left(\text{Diag}\left(X_n^{-1}AX_n\right) - \Lambda_n\right), \text{ and } X_{n+1}(t) = X_n + \Delta t X_n F\left(X_n, \Lambda_n\right).$$

This approach works only for small matrices (Example 2.2.1, and 2.2.2). If the size of a matrix is large, then to achieve the convergence in the results, we need to take a small stepsize. However with a small stepsize, too many iterations are required to obtain a few digits accuracy in the results (Example 2.2.3).

In the iterative methods, discussed in Section 2.3, and 2.6, the results do not start converging until we finish almost one half of the total iterations required. So in these iterative methods to avoid numerical instability, we need a small stepsize in each iteration at the beginning, whereas after the results start converging, we can take a large stepsize in each iteration to speed up the convergence in the results. Hence in the Differential Equation Approach to Eigencomputations with Armijo's Stepsize, we vary stepsize in each iteration. It turns out that this method does not work for all matrices. For example, if a matrix has multiple eigenvalues, or the size of the matrix is large, then the method fails (Example 2.3.2). In all of these cases, two 1×1 diagonal blocks approach each other, but as the iterations continue, the rate at which they approach each other gets slower and slower. However if by another method, a block diagonalization $A = X\Lambda X^{-1}$ can be determined (for example, using the QR method

with double implicit shift to A, we can find a real Schur form $A = QRQ^T$. Then applying Algorithm 2.2.1 to R, we can find a block diagonalization $R = P\Lambda P^{-1}$. Hence $A = X\Lambda X^{-1}$ is the block diagonalization of A, where $X = QP$,) then X can be used as the starting guess invertible matrix, and Λ can be used as the starting guess block diagonal matrix to obtain a block diagonalization of a perturbed matrix $A + \epsilon E$ with very rapid convergence, where E is an arbitrary matrix, and ϵ is a small number.

Since all matrices A are not diagonalizable, so in Section 2.5, we discuss the factors sensitivity in the Schur decomposition relative to perturbations in the coefficient matrix to compute a block Schur decomposition of a matrix A. In Section 2.6, we present an algorithm to compute a block Schur decomposition of a matrix A. This algorithm is based on sensitivity results for the factors in the Schur decomposition relative to perturbations in the coefficient matrix, and Armijo's rule from optimization theory. Here one needs a starting guess unitary matrix S_0, and a starting guess upper triangular matrix U_0, whose diagonal entries approximate the eigenvalues of the matrix A. If we take $S_0 = I$, and a diagonal matrix, whose diagonal elements are uniformly distributed points on a circle in the complex plane, which includes all Gerschgorin disks for A as U_0, then the algorithm works for all matrices. Once a block Schur decomposition $A = QUQ^H$ is computed, then this algorithm can be used to find a block Schur decomposition of a slightly perturbed matrix $A + \epsilon E$ with U as the starting guess upper triangular matrix, and Q as the starting guess unitary matrix with very rapid convergence, where E is an arbitrary matrix, and ϵ is a small number. In Section 2.7, we present some node codes which, along with some modifications to Algorithm 2.6.1, can be used in a parallel computer, where processors are set up in a ring.

In Chapter 3, we discussed how to exploit the symmetry of a matrix in the diagonalization process. In Section 3.1, we showed how to modify the diagonalization method, discussed in Chapter 1. We introduced Armijo's stepsize in the modified method to make it more practical. This method works for all symmetric matrices with distinct eigenvalues. In Section 3.2, we also modified the block diagonalization method, which is discussed in Section 2.1. Again, Armijo's stepsize is introduced to make the method practical. If at least two diagonal elements of a symmetric matrix A are equal, and the off diagonal elements are large compared to the diagonal elements, then we can not apply this method directly to A. In this case first, we reduce A to a symmetric tridiagonal matrix H using the Hessenberg reduction to A, and then apply the algorithm to H.

REFERENCES

[Atk89] K. E. Atkinson. *An Introduction to Numerical Analysis, 2nd Edition,* John Wiley & Sons, New York, NY. 1989

[Gol90] G. H. Golub, and C. F. Van Loan. *Matrix Computations, 2nd Edition,* The Johns Hopkins University Press, Baltimore, MD. 1990

[Hag87] W. W. Hager. "Bidiagonalization and Diagonalization," *Comput. Math. Applic.* Vol. 14, No. 7, 561 - 72. 1987

[Hag88] W. W. Hager. *Applied Numerical Linear Algebra,* Prentice-Hall, Englewood Cliffs, NJ. 1988

[Hel83] G. Helzer. *Applied Linear Algebra with APL,* Little, Brown and Company, Boston, MA. 1983

[Hen74] P. Henrici. *Applied and Computational Complex Analysis, Vol. I,* John Wiley & Sons, New York, NY. 1974

[Hou64] A. S. Householder. *The Theory of Matrices in Numerical Analysis,* Blaisdell Publishing Company, New York, NY. 1964

[Kal80a] R. Kalaba, K. Spingarn, and L. Tesfatsion "A New Differential Equation Method for Finding the Perron Root of a Positive Matrix," *Appl. Math. and Comp.* 7, 187 - 93. 1980

[Kal80b] R. Kalaba, K. Spingarn, and L. Tesfatsion "Individual Tracking of an Eigenvalue and Eigenvector of a Parameterized Matrix," *Nonlin. Anal. Theor. Methods and Applic.* Vol. 5, No. 4, 337 - 40. 1980

[Kal81] R. Kalaba, K. Spingarn, and L. Tesfatsion "Variational Equations for the Eigenvalues and Eigenvectors of Nonsymmetric Matrices," *J. Optim. Theor. and Applic.* Vol. 33, No. 1, 1 - 8. 1981

[Lan85] P. Lancaster, and M. Tismenetsky. *The Theory of Matrices with Applications, 2nd Edition,* Academic Press, Orlando, FL. 1985

[Nob69] B. Noble. *Applied Linear Algebra,* Prentice-Hall, Englewood Cliffs, NJ. 1969

[Rud87] W. Rudin. *Real and Complex Analysis, 3rd Edition,* McGraw-Hill Inc., New York, NY. 1987

[Ruh70] A. H. Ruhe. "An Algorithm for Numerical Determination of the Structure of a General Matrix," *BIT 10,* 196 - 216. 1970

[Ste73] G. W. Stewart. *Introduction to Matrix Computations,* Academic Press, New York, NY. 1973

[Ste76] G. W. Stewart. "Algorithm 506 : HQR3 and EXCHNG : Fortran Subroutines for Calculating and ordering the Eigenvalues of a Real Upper Hessenberg Matrix," *ACM Trans. Math. Soft.* Vol. 2, No. 3, 275 - 80. 1976

[Ste90] G. W. Stewart, and Ji-guang Sun. *Matrix Perturbation Theory,* Academic Press, San Diego, CA. 1990

[Wil65] J. H. Wilkinson. *The Algebraic Eigenvalue Problem,* Clarendon Press, Oxford, 1965

BIOGRAPHICAL SKETCH

Since my school days, I have been fascinated by mathematics and science. I completed my high school certificate in first division from Sarabari High School, Darrang, India. Then I selected the mathematics and science stream for my pre-university degree and completed in the second division from Mangaldai College, Mangaldai, India. I was not sure what career to choose when I was a high school and a pre-university student. But after I finished a pre-university degree, I decided to pursue an academic career. Subsequently, I pursued the undergraduate program in mathematics with honors in Arya Vidyapeeth College, Guwahati, India, and I completed my bachelor's degree in the second class.

Thereafter, in an endeavor to move closer to my ambition, I enrolled in the graduate program in mathematics at Gauhati University, Guwahati, India. I completed my master's degree with the first rank in the department. After being exposed to scientific principles in the undergraduate and graduate curriculums, I decided to carry out my advanced study in one of the prestigious universities of the United States. Iowa State University, in Ames, Iowa, offered me such an opportunity. After I enrolled in the graduate program in the Department of Mathematics, I felt that my background was not strong enough to pursue a Ph.D. degree. So I continued my study as a student in the Master of Science program and got my MS degree in mathematics.

Then I decided to change school for a Ph.D. degree, and the University of Southern California accepted me as a Ph.D. student. But for some personal reasons I was not comfortable at the University of Southern California and so I transferred to the University of Florida. After I enrolled as a Ph.D. student in the Department of Mathematics at the University of Florida, I got the opportunity to meet Professor

William Hager. After he agreed to supervise my research, we did further research on one of his papers, which examined the application of the eigenvalues and eigenvectors sensitivity in the eigencomputations. We succeded in creating some algorithms which use the above ideas to compute the eigenvalues and eigenvectors of a matrix.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

William W. Hager, Chair
Professor of Mathematics

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

James E. Keesling
Professor of Mathematics

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Kermit N. Sigmon
Associate Professor of Mathematics

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Bruce H. Edwards
Professor of Mathematics

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Donald W. Hearn
Professor of Industrial and Systems Engineering

This dissertation was submitted to the Graduate Faculty of the Department of Mathematics in the College of Liberal Arts and Sciences and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of the Doctor of Philosophy.

December, 1993

Dean, Graduate School